ISSN 0542-9951

P. 1877/94

5 1994

informatyka

WYDAWNICTWO SIGMA NOT

Cena 45 000 zł

informatyka

nr 5 1994
Maj
Rok wydania XXX

Miesięcznik
ISSN 0542-9951
INDEKS 36124

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
prof. dr hab. Jan GOLIŃSKI
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
mgr Krystyna
KARWICKA-RYCHLEWICZ
Władysław KLEPACZ
(redaktor naczelny)
mgr inż. Jan RYŻKO
dr Witold STANISZKIS
dr hab. Kazimierz SUBIETA
dr Zdzisław SZYJEWSKI

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Ratuszowa 11
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Mazowiecka 12 telefon: 27-43-66 telefaks: 26-80-16 teleks: 814877

W numerze:

	Strona
Budzenie świadomości środowiska – wywiad z dr. Romanem Dolczewskim – *Krystyna Karwicka*	1
Modelowanie powierzchni siatkowych w systemie AutoCAD – *Jacek Tobiasz*	2
PCN – system rozproszonego przetwarzania – *Grzegorz Czajkowski, Krzysztof Zieliński*	8
Progress 4GL/RDBMS	13
Budowa prototypu systemu ekspertowego – *Marek Miłosz, Robert Pranagal*	14
Metoda FUNCTION POINTS – *Beata Czarnacka-Chrobot*	19
Ze świata	
Główni wytwórcy komputerów osobistych – *Jan Ryżko*	25
CeBIT '94 Refleksje – *Władysław Klepacz*	28

W najbliższych numerach:

- **Krystyna Karwicka-Rychlewicz rozmawia z dr. Wacławem Iszkowskim, Prezesem Polskiej Izby Informatyki i Telekomunikacji.**
- **Halina Kwaśnicka i Urszula Markowska-Kaczmar zajmują się problematyką systemów eksperckich pracujących w czasie rzeczywistym z wykorzystaniem logiki rozmytej.**
- **Marek Gajęcki i Krzysztof Zieliński prezentują PVM – system oprogramowania do realizacji rozproszonych programów heterogenicznych w sieciach maszyn sekwencyjnych, równoległych i wektorowych.**
- **Anna Ławrynowicz opisuje ekspertowy system wspomagania sterowania produkcją.**
- **Marek Stabrowski przedstawia implementację mechanizmu definiowania, kompilacji, konsolidacji i wykonywania podprogramów w systemie symulacyjnym AMIL.**

Warunki prenumeraty na 1994 r.

Zamówienia na prenumeratę czasopism wydawanych przez Wydawnictwo SIGMA-NOT można składać w dowolnym terminie. Mogą one obejmować dowolny okres czasu, tzn. dotyczyć dowolnej liczby kolejnych zeszytów każdego czasopisma.

Zamawiający może otrzymywać zaprenumerowany przez siebie tytuł począwszy od następnego miesiąca po dokonaniu wpłaty. Zamówienia na zeszyty sprzed daty otrzymania wpłaty będą realizowane w miarę możliwości – z posiadanych zapasów magazynowych.

Warunkiem przyjęcia i realizacji zamówienia jest otrzymanie z banku potwierdzenia dokonania wpłaty przez prenumeratora. Dokument wpłaty jest równoznaczny ze złożeniem zamówienia.

Wpłat na prenumeratę można dokonywać na ogólnie dostępnych blankietach w Urzędach Pocztowych (przekazy pieniężne) lub Bankach (polecenie przelewu), przekazując środki pod adres:
Wydawnictwo SIGMA-NOT Spółka z o.o.
Zakład Kolportażu
00-716 Warszawa, skr. poczt. 1004
konto:
PBK S.A. III 0/Warszawa nr 370015-1573-139-11

Wpłaty na prenumeratę od marca br. przyjmują także wszystkie urzędy pocztowe nadawczo-odbiorcze oraz doręczyciele na terenie całego kraju

Na blankiecie wpłaty należy czytelnie podać nazwę zamawianego czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty oraz własny adres.
Na życzenie prenumeratora, zgłoszone np. telefonicznie, Zakład Kolportażu ul. Bartycka 20, 00-950 Warszawa, (telefony: 40-30-86, 40-35-89 oraz 40-00-21 wew. 249, 293, 299) wysyła specjalne blankiety zamówień wraz z aktualną listą tytułów i cennikiem czasopism.

Odbiorcy zagraniczni mogą otrzymywać czasopisma poprzez prenumeratę dewizową (wpłata dokonywana poza granicami Polski w dewizach, wg cennika dewizowego z cenami podanymi w dolarach amerykańskich) lub poprzez zamówioną w kraju prenumeratę ze zleceniem wysyłki za granicę (zamawiający podaje dokładny adres odbiorcy za granicą, dokonując równocześnie wpłaty w wysokości dwukrotnie wyższej niż cena normalnej prenumeraty krajowej).

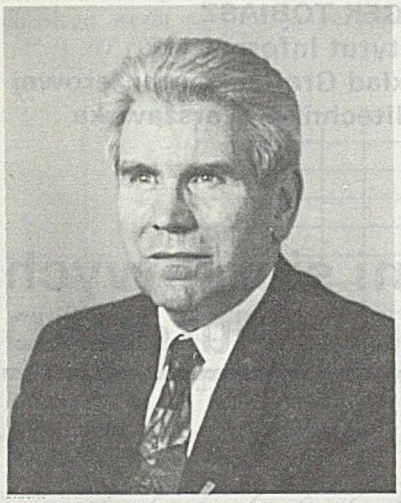
Egzemplarze archiwalne (sprzedaż przelewowa lub za zaliczeniem pocztowym) można zamawiać pisemnie, kierując zamówienia pod adresem: Wydawnictwo SIGMA NOT, Spółka z o.o. Zakład Kolportażu, 00-716 Warszawa, ul. Bartycka 20, paw. B, tel. 40-37-31, natomiast za gotówkę można je nabyć w Klubie Prasy Technicznej w Warszawie ul. Mazowieckiej 12, tel. 26-80-17.

Istnieje możliwość zaprenumerowania 1 egz. czasopisma po cenie ulgowej przez indywidualnych członków stowarzyszeń naukowo-technicznych zrzeszonych w FSNT oraz przez uczniów zawodowych i studentów szkół wyższych. Blankiet wpłaty na prenumeratę ulgową musi być opatrzony na wszystkich odcinkach pieczęcią koła SNT lub szkoły.

W przypadku zmiany cen w okresie objętym prenumeratą Wydawnictwo zastrzega sobie prawo do wystąpienia o dopłatę różnicy cen oraz prawo do realizowania prenumeraty tylko w pełni opłaconej.
Cena jednego egzemplarza: normalna 45 000 zł, ulgowa 31 500 zł
Wartość prenumeraty w zł:
Normalna: kwartalna 135 000, półroczna 270 000, za trzy kwartały 404 000
Ulgowa: kwartalna 94 500, półroczna 189 000, za trzy kwartały 283 500

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 126/94

P.1877/94



POLITECHNIKA ŚLĄSKA Biblioteka Główna w Gliwicach

Budzenie świadomości środowiska

Nagrodę INFO-STAR '93 w dziedzinie – Popularyzacja Informatyki – otrzymał w tym roku dr Roman Dolczewski – współzałożyciel i pierwszy prezes (obecnie wiceprezes) Stowarzyszenia Polski Rynek Oprogramowania „PRO", a zarazem prezes firmy CSBI. Z laureatem rozmawia Krystyna Karwicka

– Laur INFO-STAR '93 otrzymał Pan za – jak brzmi oficjalne oświadczenie jury – „całokształt dokonań dla rozwoju informatyki w Polsce ze szczególnym uwzględnieniem osiągnięć w 1993 r.". Jakie są te osiągnięcia?

– Myślę, że INFO-STAR dla mnie to wyraz uznania dla całej tej grupy osób i firm, które przyczyniły się do budowania świadomości polskiego środowiska informatycznego. Zrozumiało ono, że trzeba się zorganizować i bronić interesów twórców oprogramowania i jego legalnych dystrybutorów. Dzięki temu udało się nam doprowadzić, razem z innymi grupami twórców, do uchwalenia ustawy o Prawie Autorski. Już w 1991 r. Esther Dyson, znana „guru" amerykańskiej informatyki, spytała się na jednej z międzynarodowych konferencji, dlaczego w Polsce nie ma organizacji zrzeszającej twórców oprogramowania. Wprawdzie od lat działa Polskie Towarzystwo Informatyczne, ale to jest w zasadzie organizacja zawodowa informatyków, w większości użytkowników oprogramowania, i nie może reprezentować specyficznych interesów producentów oprogramowania. Stąd też narodził się pomysł utworzenia Stowarzyszenia Polski Rynek Oprogramowania „PRO". Okazało się to trudniejsze niż przypuszczaliśmy na początku. Wprawdzie mamy ok. 2 tys. firm komputerowych, w tym co najmniej 400 softwerowych, ale nie było wśród nich gwałtownej reakcji na propozycje zorganizowania stowarzyszenia, które pilnowałoby spraw i interesów twórców softweru. We wrześniu 1992 r., po roku pracy przygotowawczej, udało się zarejestrować Stowarzyszenie, w czerwcu 1993 r. odbyło się walne zgromadzenie członków. Członkami stowarzyszenia jest ok. 50 osób, finansowo zaś wspiera naszą działalność ok. 20 firm.

– W stosunku do liczby polskich domów softwerowych to niewiele...

– To prawda, ale mimo to reprezentujemy, jak sądzę, interesy całego środowiska. Sądzę też, że nasze działania ułatwiły powstanie Polskiej Izby Informatyki i Telekomunikacji (PIIiT), narodziła się bowiem świadomość, że trzeba się zorganizować i wspólnie bronić.

– W jakim stopniu Pana inicjatywa wynikła z potrzeby bronienia interesów CSBI, czyli firmy, którą Pan kieruje?

– CSBI w zasadzie nie ma kłopotów z nielegalnym kopiowaniem programów. Naszego czołowego produktu – BANKIERĄ – nie da się uruchomić bez środowiska Progress, które stwarza odpowiednie zabezpieczenia. Ukradzenie go w prosty sposób nie jest możliwe. Moje działania nie wynikają więc z potrzeby ochrony własnych interesów, ale ze świadomości, że jesteśmy częścią tego środowiska i jeżeli nie uporamy się z problemem piractwa, to żadna firma softwerowa nie będzie mogła prawidłowo funkcjonować i rozwijać się. Z naszymi specjalistycznymi systemami współpracuje zresztą cała gama oprogramowania standardowego.

– Wiadomo powszechnie, że ustawa o Prawie Autorskim nie satysfakcjonuje twórców polskiego oprogramowania. Dlaczego doszło do jej uchwalenia w obecnym kształcie?

– Kiedy wreszcie udało się powołać do życia Stowarzyszenie „PRO" okazało się, że tak naprawdę to jesteśmy co najmniej o rok spóźnieni, Komisja Sejmowa pracowała już bowiem nad ustawą. Działając wspólnie z PTI, Stowarzyszeniem Rozwoju Systemów Otwartych i PIIiT udało nam się przynajmniej tak zmienić projekt ustawy, by była ona zgodna z zaleceniami Rady Europejskiej Wspólnoty Gospodarczej z maja 1991 r. Niestety, nie udało się usunąć z projektu ustawy lub chociażby zmienić zapisu, iż darowuje się wszystkie kradzieże oprogramowania, które powstało przed uchwaleniem Prawa Autorskiego. Pozycja użytkowników okazała się silniejsza od pozycji twórców. Ale nikt w Sejmie nie może powiedzieć, że nie widział o co chodzi, ponieważ Komisje Sejmowe pracujące nad projektem Ustawy otrzymały nasze wystąpienia w tej sprawie. Uczestniczyliśmy także w niektórych posiedzeniach. Gdyby Stowarzyszenie powstało rok wcześniej, gdybyśmy odpowiednio wcześniej zaczęli kształtować świadomość prawodawców i członków Komisji Sejmowej – może nie doszłoby do tego dramatu.

– Czy to nie za mocne określenie sytuacji, jaka się wytworzyła na skutek takiego a nie innego sformułowania przepisu o abolicji?

– Sądzę, że to jest dramatyczna sytuacja. Nasi legislatorzy jak gdyby uznali, że w Polsce nie ma firm i domów softwerowych, że nie tworzyło się u nas oprogramowania, że to co dotychczas powstało, nie ma żadnej wartości. Można więc wszystko spokojnie, bez opłat kopiować, zmieniać, używać w dowolny sposób. Na giełdzie kpią sobie z twórców polskiego oprogramowania. Tymczasem to głównie dzięki polskiemu oprogramowaniu funkcjonują systemy bankowe, wiele innych instytucji użyteczności publicznej i firm. Mimo to dobrze, że w ogóle jest ustawa, satysfakcjonuje ona firmy zagraniczne, co ma oczywiście znaczenie dla całego rynku oprogramowania.

– Czy Stowarzyszenie i inne organizacje środowiskowe będą walczyć o zmianę kształtu ustawy?

– Jeśli chodzi o zapis abolicyjny nasi pracownicy uważają, że można wystąpić w tej sprawie do Trybunału Stanu czy do

dokończenie na s. 7

JACEK TOBIASZ
Instytut Informatyki
Zakład Grafiki Komputerowej
Politechnika Warszawska

Modelowanie powierzchni siatkowych w systemie AutoCAD

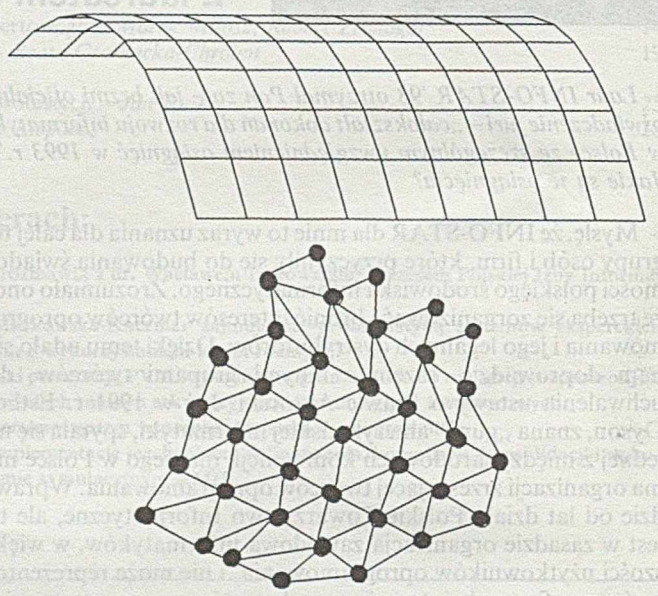
AutoCAD, produkt firmy Autodesk Inc., jest klasycznym programem komputerowym wspomagającym projektowanie. Ze względu na swoje szerokie możliwości znajduje zastosowanie w takich dziedzinach inżynierskich, jak: mechanika, architektura, budownictwo czy topografia. W każdym z tych zastosowań jest wykorzystywana graficzna prezentacja obiektów dwu- i trójwymiarowych. AutoCAD pozwala w szczególności tworzyć modele wykorzystywane w grafice komputerowej i dlatego będzie dalej określany terminem *modeler*. Modele takie mogą być wykorzystywane w symulacjach z uwzględnieniem czynników fizycznych [6].

Program został zaimplementowany w różnych środowiskach sprzętowo-programowych (MS-DOS, UNIX, VMS), co daje możliwość bezpośredniej współpracy różnych systemów.

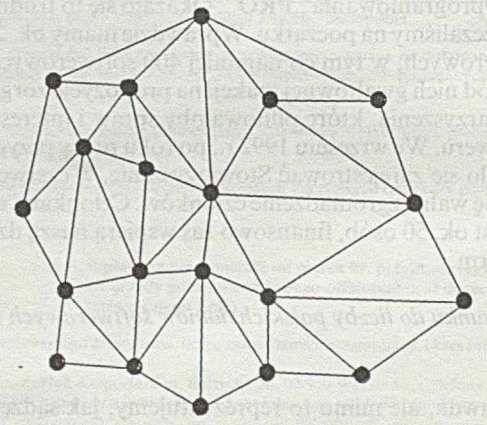
Poniżej przedstawiono krótką charakterystykę programu AutoCAD istotną ze względu na zastosowania w grafice [2].

Program zawiera narzędzia tworzenia modeli dwu- i trójwymiarowych. Wielkości charakteryzujące model są niezależne od zasobów sprzętowych komputera (rozdzielczość grafiki itd.) – rozmiar, skala rysunku itp. są definiowane przez użytkownika. Istnieje możliwość tworzenia bibliotek symboli i elementów oraz możliwość adaptacji programu do specjalizowanych zastosowań użytkownika (np. AutoLISP, system ADS dla AutoCAD-a od wersji 11).

W artykule zostaną przedstawione zagadnienia wykorzystania programu AutoCAD do modelowania powierzchni siatkowych. Proces modelowania powierzchni może stanowić jeden z pierwszych elementów dużego systemu wykorzystującego metody symulacji zjawisk fizycznych, realistycznego modelowania rzeczywistości. W dalszej części zostaną zaprezentowane narzędzia modelowania dostępne w systemie AutoCAD oraz właściwości środowiska pozwalające na wykorzystanie adaptacyjnych funkcji modelera.



Rys. 1. Przykład siatek o topologii czworokątnej i sześciokątnej



Rys. 2. Przykład siatki o niejednorodnej topologii

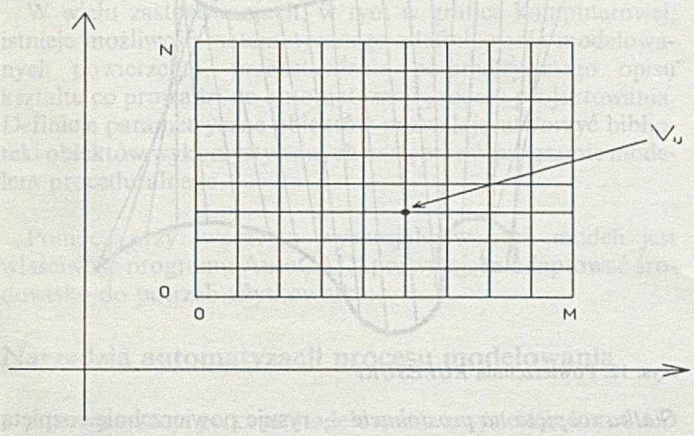
Powierzchnie siatkowe

Dla potrzeb artykułu przyjmiemy, że powierzchnia siatkowa jest to zbiór węzłów połączonych krawędziami (liniami prostymi). Dla każdego węzła siatki możemy określić jego współrzędne kartezjańskie x, y, z oraz zbiór węzłów z nim sąsiadujących. W zależności od liczby punktów sąsiadujących z każdym z wierzchołków oraz sposobu określenia sąsiedztwa, możemy wyróżnić siatki o topologii trójkątnej, czworokątnej itd. (rys. 1). W przypadku różnej liczby węzłów sąsiednich otrzymujemy siatkę niejednorodną (rys. 2) [7, 8].

Siatka o topologii czworokątnej ma prostą strukturę, pozwalającą na łatwy opis powierzchni. Każdy węzeł jest jednoznacznie określony przez parę indeksów i, j (rys. 3).

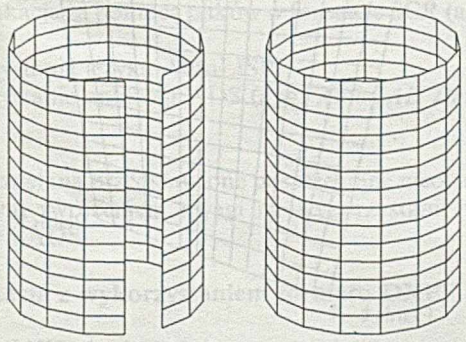
Definicja siatki wymaga określenia współrzędnych kartezjańskich każdego węzła x, y, z; węzły sąsiadujące są zdeterminowane strukturą czworokątną siatki – każdy węzeł wewnętrzny ma czterech sąsiadów, a węzły brzegowe – dwóch albo trzech. Dzięki takiej budowie siatki możemy w łatwy sposób wprowadzić opis parametryczny powierzchni, gdzie współrzędne

każdego węzła są funkcją pary indeksów i, j: $V_{i,j} = (x(i,j), y(i,j), z(i,j))$.



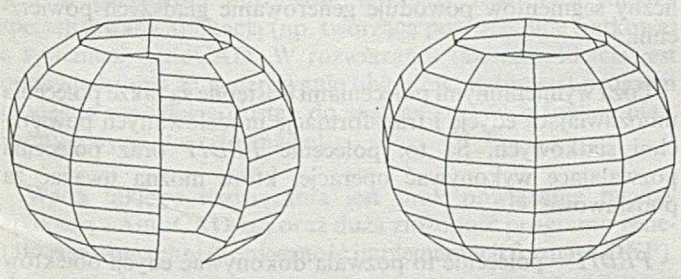
Rys. 3. Siatka o topologii czworokątnej

Poniżej podano przykłady wykorzystania siatki (o topologii prostokątnej) do modelowania cylindrów, kul i torusów.



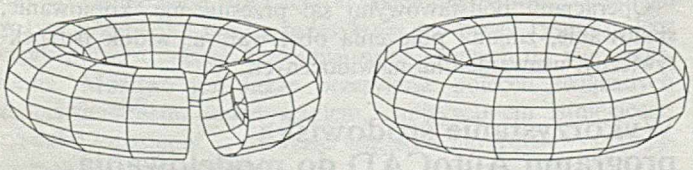
Rys. 4. Siatka cylindryczna „nie domknięta" i „domknięta"

Cylinder (rys. 4) powstaje w wyniku domknięcia siatki prostokątnej według jednego z indeksów. Zamknięcie polepsza regularność siatki (eliminacja dwóch krawędzi).



Rys. 5. Siatka kulista – dwie fazy definiowania

Kula (rys. 5) powstaje w wyniku domknięcia siatki prostokątnej według obu indeksów.



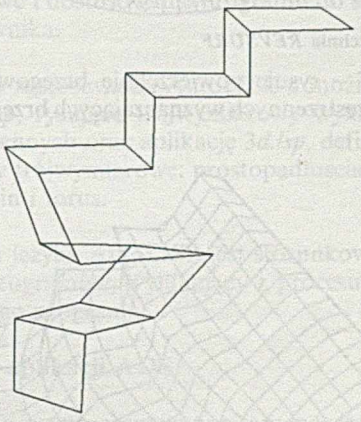
Rys. 6. Siatka torusowa

Torus (rys. 6) powstaje w wyniku domknięcia siatki prostokątnej według obu indeksów, jednak jeśli w przypadku kuli możemy określić modyfikację powierzchni jako domknięcie do wewnątrz, to w przypadku torusa mamy do czynienia z domknięciem do wewnątrz według jednego indeksu i na zewnątrz według drugiego.

Narzędzia modelowania przestrzennego w AutoCAD wersja 11

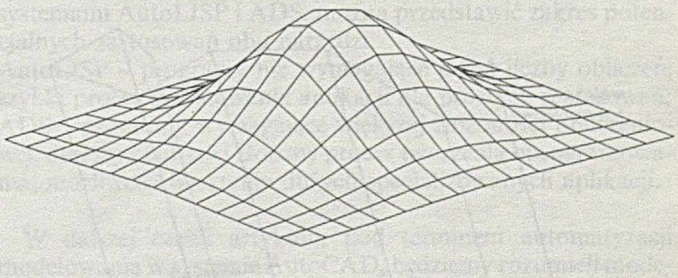
Poniżej zostaną przedstawione podstawowe polecenia umożliwiające modelowanie przestrzenne – w szczególności modelowanie powierzchni siatkowych. Polecenia te stanowią standardowy zbiór poleceń programu AutoCAD i mają swoje odpowiedniki w innych modelerach. Są to:

- *3DFACE* – rysuje obszar *3D* na podstawie punktów leżących na obwodzie. Segmenty obszaru definiuje się za pomocą czterech punktów brzegowych (segmenty czworokątne). Każdy segment obszaru *3D* sąsiaduje tylko z dwoma innymi segmentami (ograniczenie definiowanych powierzchni do struktur typu „schody" – rys. 7). Za pomocą obszarów 3D można tworzyć modele ściankowe.



Rys. 7. Struktura *3DFACE*

- *3DMESH* – rysuje powierzchnie siatkowe o czworokątnej topologii segmentów. Siatka taka ma określone rozmiary M i N oraz współrzędne każdego węzła $v_{i,j}$. Polecenie to umożliwia modelowanie złożonych powierzchni funkcyjnych (funkcja dwu zmiennych i, j) lub parametrycznych (opis parametryczny definiowany przez projektanta). Siatka taka jest zbudowana z wielu wzajemnie połączonych obszarów *3DFACE* (rys. 8).



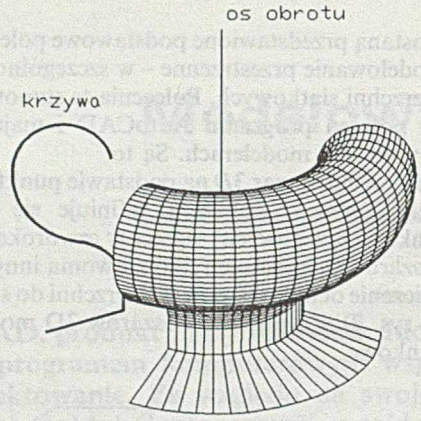
Rys. 8. Struktura *3DMESH*

- *PFACE* – rysuje powierzchnie siatkowe o dowolnej strukturze topologicznej. Definicja powierzchni wymaga określenia dla każdego wierzchołka jego współrzędnych oraz sąsiadujących z nim węzłów. Polecenie to daje możliwość tworzenia siatek niejednorodnych.

Przedstawione narzędzia pozwalają na specyfikację powierzchni siatkowej przez określenie wszystkich należących do niej punktów. AutoCAD oferuje również możliwość definiowania

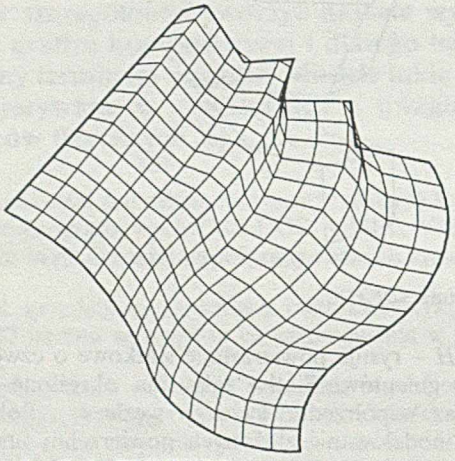
powierzchni (automatycznej generacji węzłów) na podstawie określonych elementów konstrukcyjnych. Służą do tego następujące polecenia:

REVSURF – rysuje powierzchnię obrotową przez obrót krzywej profilowej wokół osi obrotu o określony kąt (rys. 9);



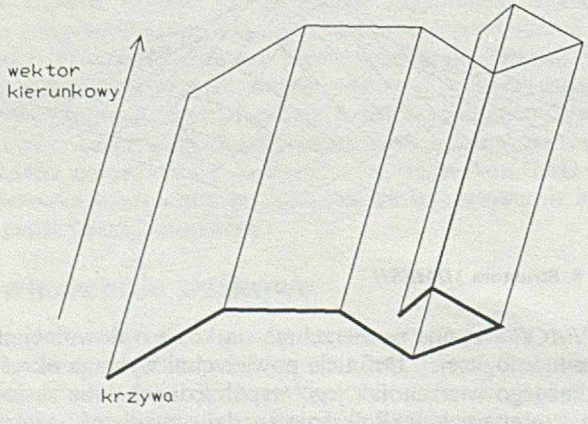
Rys. 9. Powierzchnia *REVSURF*

EDGESURF – rysuje powierzchnię brzegową na podstawie krzywych przestrzennych wyznaczających brzeg tej powierzchni (rys. 10);



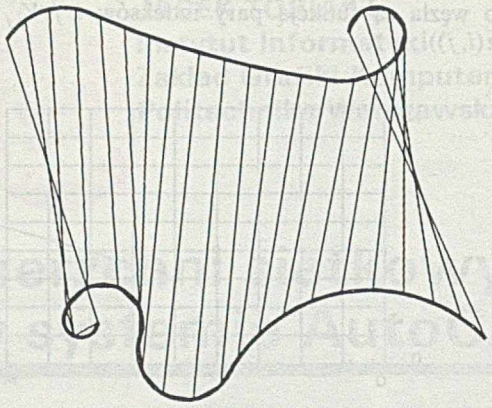
Rys. 10. Powierzchnia *EDGESURF*

TABSURF – rysuje powierzchnię równoległą na podstawie krzywej definiującej i wektora kierunkowego (rys. 11);



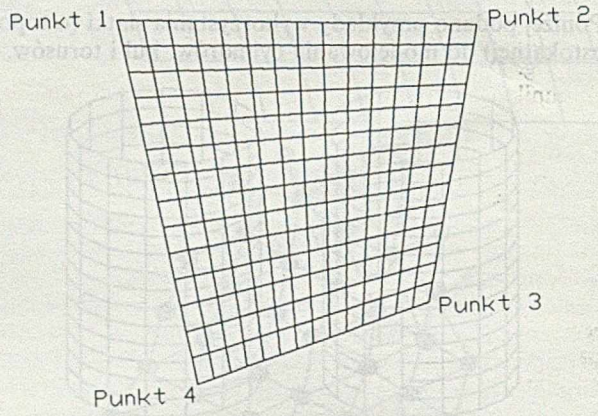
Rys. 11. Powierzchnia *TABSURF*

RULESURF – rysuje powierzchnię rozpiętą między dwoma elementami konstrukcyjnymi (rys. 12).



Rys. 12. Powierzchnia *RULESURF*

Siatka rozpięta na prostokącie — rysuje powierzchnię rozpiętą między czterema wskazanymi punktami (rys. 13).



Rys. 13. Powierzchnia rozpięta między czteroma punktami

Przedstawione powierzchnie, podobnie jak siatki, są zbudowane z wielu obszarów 3*D* (3*DFACE*). Dla każdej powierzchni można określić liczbę tworzących ją segmentów prostokątnych (co odpowiada rozmiarom *M, N* siatki 3*DMESH*). Zwiększenie liczby segmentów powoduje generowanie gładszych powierzchni.

Poza wymienionymi poleceniami dostępne są także polecenia umożliwiające edycję i transformację modelowanych powierzchni siatkowych. Są to: polecenie *PEDIT* oraz polecenia pozwalające wykonywać operacje, które można uważać za podstawowe.

PEDIT – polecenie to pozwala dokonywać edycji obiektów wygenerowanych za pomocą wymienionych wyżej funkcji. Istnieje możliwość modyfikowania położenia wierzchołków, wygładzania (krzywe Bèzier, B-spline) oraz domknięcia powierzchni wzdłuż określonego kierunku.

Operacjami podstawowymi są: przesuwanie, kopiowanie, skalowanie, zmiana położenia obserwatora, widok perspektywiczny, usuwanie linii niewidocznych.

Wykorzystanie środowiska programu AutoCAD do modelowania powierzchni siatkowych

Polecenia przedstawione powyżej należą do standardowych narzędzi dostępnych w programach różnych modelerów. Cechą charakterystyczną modelowania wykorzystującego przedsta-

wione komendy jest konieczność interakcyjnej współpracy projektanta z programem.

W wielu zastosowaniach, w tym w grafice komputerowej, istnieje możliwość matematycznego zdefiniowania modelowanych powierzchni, przedstawienia parametrycznego opisu kształtu co prowadzi do automatyzacji procesu projektowania. Definicje parametryczne obiektów pozwalają utworzyć biblioteki obiektów wykorzystywanych następnie w programie modelera proceduralnego.

Pomocą przy tworzeniu wyspecjalizowanych modeli jest właściwość programu AutoCAD, pozwalająca adaptować środowisko do potrzeb użytkownika.

Narzędzia automatyzacji procesu modelowania

Automatyzacja procesu modelowania w systemie AutoCAD jest możliwa dzięki wykorzystaniu jednego z czterech mechanizmów. Są to:

- komunikacja z AutoCAD-em za pomocą plików w formacie DXF (ang. *Drawing Exchange Format*),
- komunikacja za pomocą plików w formacie SCR (ang. *Script Files*),
- wykorzystanie języka AutoLISP,
- wykorzystanie aplikacji ADS (ang. *AutoCAD Development System*).

Poniżej zostaną przedstawione poszczególne mechanizmy, ze szczególnym zwróceniem uwagi na wykorzystanie systemów AutoLISP i ADS.

Komunikacja z wykorzystaniem formatu DXF

Format DXF został zdefiniowany w celu ułatwienia komunikacji AutoCAD-a ze światem zewnętrznym i stanowi standard oferowany przez większość programów wspomagających modelowanie.

Proces automatyzacji w systemie AutoCAD wykorzystuje w tym przypadku możliwość wymiany informacji między specjalizowaną aplikacją (np. tworzącą powierzchnie siatkowe) a systemem AutoCAD. W rozwiązaniu takim, aplikacja jest odpowiedzialna za generowanie obiektów, natomiast program modelera umożliwia dostęp do zaawansowanych funkcji platformy (edycja, transformacje).

Wadą takiego rozwiązania jest brak powiązania między aplikacją i AutoCAD-em oraz duża złożoność programu generatora powierzchni (konieczność implementacji formatu DXF).

Komunikacja z wykorzystaniem zbiorów SCR

Rozwiązanie to jest podobne do poprzedniego, nie wymaga jednak transferu danych w formacie DXF. Komunikacja między specjalizowaną aplikacją a systemem AutoCAD polega w tym przypadku nie na przesyłaniu gotowego rysunku, lecz ciągu poleceń (zleceń kreślenia) prowadzących do jego uzyskania. Możliwe jest także przesyłanie poleceń związanych z wyświetlaniem, wczytywaniem, zapisywaniem istniejącego rysunku – dostępne są wówczas wszystkie narzędzia występujące w programie modelera.

Rozwiązanie to ma identyczne wady jak komunikacja z wykorzystaniem formatu DXF oraz cechuje się długim czasem wykonywania zbioru zleceń SCR. Może być wykorzystywane do sterowania AutoCAD-em w trybie wsadowym.

Wykorzystanie języka AutoLISP

AutoLISP jest językiem programowania będącym specjalizowaną odmianą LISP-u, dostosowaną do operowania na obiektach AutoCAD-a [3, 4]. Interpreter AutoLISP-u jest zintegrowany z AutoCAD-em, co umożliwia automatyzację procesu modelowania bezpośrednio w środowisku programu. Dzięki AutoLISP-owi istnieje możliwość dodawania nowych komend, zmieniania właściwości już istniejących, programowe wykonywanie wielu złożonych obliczeń, operacji rysunkowych itd.

Program w AutoLISP-ie ma możliwość wykorzystania zarówno poleceń AutoCAD-u i funkcji edytora rysunku, jak i mechanizmów typowych dla języków programowania.

Aplikacja zapisana w AutoLISP-ie może zawierać algorytm generowania złożonych obiektów. Wyniki obliczeń mogą być przekazane do edytora rysunku AutoCAD-a. AutoLISP zawiera funkcje interakcyjnej współpracy z użytkownikiem, co pozwala na tworzenie specjalistycznych aplikacji przejmujących zadania dialogowe i dostosowanie programu do specyficznych potrzeb użytkownika.

Jako przykład zastosowania AutoLISP-u można podać aplikację *wykres.lsp*, rysującą trójwymiarowy wykres dowolnej funkcji dwu zmiennych oraz aplikację *3d.lsp*, definiującą standardowe obiekty trójwymiarowe: prostopadłościan, ostrosłup, kopułę, sferę, klin i torus.

Główną wadą języka AutoLISP jest stosunkowo długi czas wykonywania programów, wynikający z procesu interpretacji tekstu źródłowego programu.

Wykorzystanie aplikacji ADS

Aplikacja ADS jest programem napisanym w języku C, który pracuje w środowisku AutoCAD-a [1]. System ADS, podobnie jak AutoLISP, ma możliwość korzystania z bogatej biblioteki funkcji AutoCAD-a oraz ze wszystkich mechanizmów języka C.

Istotną cechą, odróżniającą aplikacje ADS i AutoLISP jest większa szybkość wykonywania programów napisanych w języku C (nie występuje faza interpretacji tekstu źródłowego) oraz sposób zapisu algorytmu w obu środowiskach. W AutoLISP-ie mamy listową reprezentację struktur danych, a w języku C – programowanie proceduralne.

Uwzględniając różnice i podobieństwa występujące między systemami AutoLISP i ADS, można przedstawić zakres potencjalnych zastosowań obu narzędzi:
AutoLISP – programy nie wymagające dużej liczby obliczeń; szybki proces powstawania aplikacji dla prostych zastosowań;
ADS – programy wymagające większej złożoności obliczeniowej; dłuższy i bardziej złożony proces tworzenia oprogramowania; możliwość tworzenia dużych specjalizowanych aplikacji.

W dalszej części artykułu, pod terminem automatyzacji modelowania w systemie AutoCAD, będziemy rozumieli modelowanie z wykorzystaniem systemów AutoLISP i ADS.

Możliwość zastosowania aplikacji ADS i AutoLISTP do modelowania powierzchni siatkowych

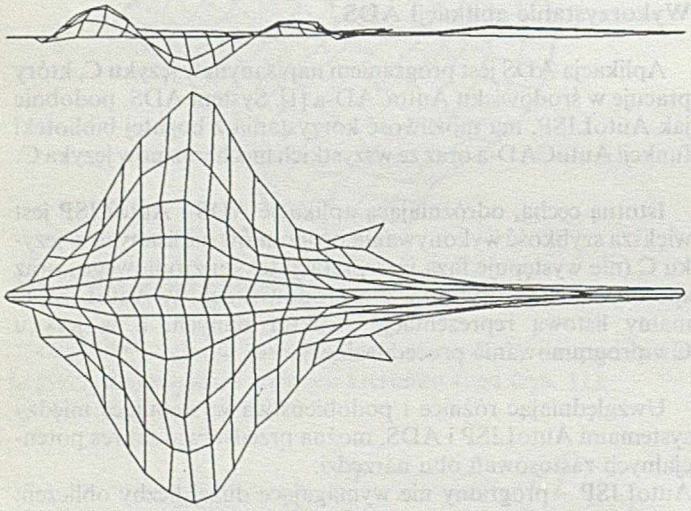
Modelowanie powierzchni siatkowych może być związane z kilkoma fazami. Są to:
a) generowanie powierzchni siatkowej – utworzenie struktury węzłów połączonych krawędziami oraz nadanie im współrzęd-

nych przestrzennych; jest to główna funkcja aplikacji w środowisku AutoCAD-a;
b) modelowanie zachowania się powierzchni (możliwość utworzenia modelu fizycznego oraz symulacja zjawisk naturalnych);
c) wizualizacja powierzchni siatkowej – prezentacja wygenerowanych obiektów, zachowania się modelu fizycznego;
d) eksportowanie wyników generowania, modelowania fizycznego poza środowisko AutoCAD-a.

Generowanie powierzchni siatkowej (faza a)

Obiekt geometryczny może zostać utworzony automatycznie na podstawie opisu zawartego w algorytmie lub danych dostarczanych z zewnątrz do aplikacji ADS (możliwość przyjęcia dowolnie zdefiniowanego formatu danych). Utworzone powierzchnie mogą być następnie poddawane operacjom transformacji i edycji wbudowanym w systemie AutoCAD-a lub zaimplementowanym przez aplikacje.

Opis powierzchni może być dostosowany do charakteru modelowanego obiektu (opis funkcyjny, różne formy parametryzacji) oraz jego definicji matematycznej. Wykorzystanie modelera proceduralnego umożliwia stworzenie szablonu powierzchni oraz bibliotek obiektów ułatwiających późniejsze modelowanie. Zdefiniowany szablon może być następnie wykorzystany w procesie parametrycznego generowania elementów składowych innych obiektów. Możliwe jest także zastosowanie programu generującego jako modułu filtrującego (przekształcającego) istniejące obiekty według określonych formuł (załamywanie, zawijanie, modulacja funkcyjna powierzchni). Na rys. 14 przedstawiono model obrazujący jedną z faz ruchu płaszczki, uzyskaną w wyniku modulacji funkcyjnej powierzchni.



Rys. 14. Przykład modulacji funkcyjnej powierzchni – model płaszczki (*Hypertremata*)

Zastosowanie aplikacji umożliwia określenie dodatkowych atrybutów związanych z modelowaną powierzchnią, np. przypisywanie masy węzłom siatki, określenie sztywności lub naprężeń występujących w krawędziach.

Dodatkowe właściwości obiektów nie mają bezpośredniego odwzorowania w systemie AutoCAD, mogą być jednak reprezentowane graficznie przez grubość lub barwę elementów rysunkowych.

Modelowanie zachowania się powierzchni (faza b)

Przypisanie określonego modelu fizycznego wymaga zakodowania jego reprezentacji w aplikacji ADS. Aplikacja taka może uwzględniać wiele parametrów powierzchni siatkowej zdefiniowanych w fazie generowania, określać środowisko właściwe dla symulowanego obiektu. Modelowanie może opierać się na fizycznych modelach układów rzeczywistych, działając całkowicie niezależnie od systemu AutoCAD [6, 7].

Wizualizacja powierzchni siatkowych (faza c)

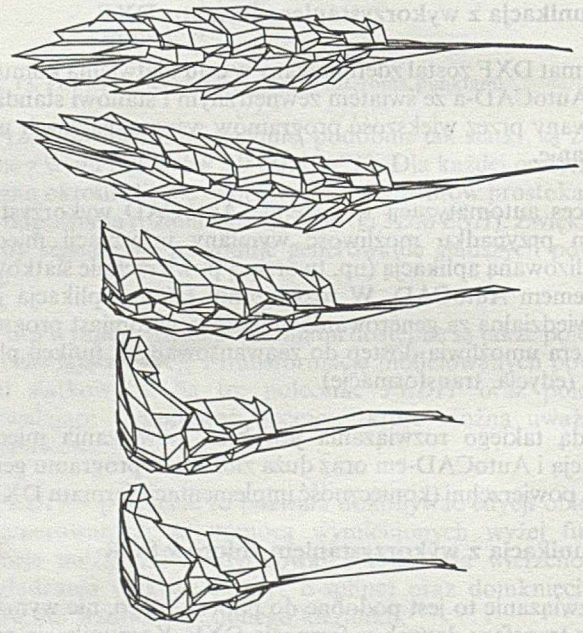
Wyniki generowania i modelowania fizycznego mogą zostać zwizualizowane przy wykorzystaniu funkcji oferowanych przez system AutoCAD. Dostępne funkcje umożliwiają prezentację obiektów trójwymiarowych z usuwaniem linii niewidocznych, z dowolnie wybranym punktem obserwacji, perspektywą. AutoCAD udostępnia aplikacji złożony interfejs graficzny użytkownika.

Ekspertowanie wyników (faza d)

Wyniki realizacji każdej z wyżej przedstawionych faz, mogą zostać przesłane w dowolnym formacie na zewnątrz systemu AutoCAD. W połączeniu z fazą (a) stwarza to możliwość konwersji niestandardowych formatów oraz dostosowanie aplikacji do innych istniejących rozwiązań.

Przykłady

Przedstawiona technika modelowania powierzchni siatkowych z wykorzystaniem środowiska AutoCAD/ADS znalazła zastosowanie w systemie modelowania powierzchni 3*D* za pomocą sieci elastycznych [6, 7]. Na rys. 15 przedstawiono kolejne fazy transformacji modelu liścia, od modelu płaskiego (z małym czynnikiem losowym) do modelu obrazującego uschnięty liść.



Rys. 15. Kolejne fazy transformacji modelu liścia

* * *

Przedstawione powyżej cztery fazy modelowania powierzchni siatkowych w systemie AutoCAD, odzwierciedlają zalety i wady użytkowania mechanizmów udostępnianych przez program AutoCAD.

Podstawową zaletą takiego rozwiązania jest adaptacja istniejącego środowiska, wykorzystanie wbudowanych funkcji-poleceń wysokiego poziomu (generowanie obiektów geometrycz-

nych, wizualizacja, transfer w formacie DXF). Podejście takie upraszcza konstruowanie oprogramowania, zmniejsza jego koszty oraz czas potrzebny do realizacji projektu.

Dodatkową zaletą jest wykorzystanie standardowych mechanizmów umożliwiających łatwą współpracę z innymi programami o podobnym charakterze.

Przyjęcie takiego rozwiązania pociąga za sobą także pewne wady: uzależnienie się od funkcji oferowanych przez środowisko AutoCAD-a, nieefektywne wykonywanie niektórych operacji (np. niemożliwa jest realizacja animacji).

Podsumowując można stwierdzić, iż wykorzystanie środowiska AutoCAD-a jest korzystne przy tworzeniu aplikacji, dla których jest wystarczający dostępny interfejs graficzny oraz wymaga się współpracy z systemami CAD. Pozwala to relatywnie tanio tworzyć oprogramowanie modelujące powierzchnie siatkowe, które może stanowić podstawę do dalszego rozwoju samodzielnych, specjalizowanych programów graficznych.

Autor pragnie podziękować dr. Cezaremu Stępniowi z Zakładu Grafiki Komputerowej Instytutu Informatyki Politechniki Warszawskiej za cenne uwagi i wskazówki pomocne przy opracowywaniu niniejszego artykułu, w którym przedstawiono część wyników uzyskanych w trakcie realizacji projektu badawczego KBN 8 8324 91 02.

LITERATURA

[1] ADS – Podręcznik programisty. Autodesk Ltd., 1991
[2] AutoCAD wersja 11 – Podręcznik użytkownika. Autodesk Ltd., 1991
[3] AutoLISP – Podręcznik programisty. Autodesk Ltd., 1991
[4] Horoczyczak M., Skierski J.: AutoLISP dla użytkowników AutoCAD-a. PLJ, Warszawa 1991
[5] Kernigham B.W., Ritchie D.M.: Język C. WNT, Warszawa 1988
[6] Stępień C., Kalisiak Z., Gracki K.: Shapes manipulation (w druku)
[7] Stępień C., Gracki K., Kalisiak Z.: Elastic Meshes with aditional connections (w druku)
[8] Stępień C., Chabko K.: The conversion of the data solids in the polygonal form into the mesh form (w druku).

Wskazówki dla Autorów

Nadsyłane artykuły nie mogą być publikowane lub przeznaczone do opublikowania w innych czasopismach.

Materiał oprócz tekstu zasadniczego powinien zawierać:
* krótki życiorys zawodowy Autora i jego zdjęcie,
* wykaz literatury,
* tabele,
* materiał ilustracyjny (rysunki, zdjęcia czarno-białe, wydruki) dołączony do artykułu (nie wklejać materiału w tekst),
* podpisy pod ilustracje.

Na osobnej stronie prosimy podać: tytuł naukowy, nazwisko i imię, nazwę zakładu pracy, numer telefonu służbowego oraz informację, jaką drogą przekazać honorarium – kasa Wydawnictwa, poczta, konto.

W związku z naliczaniem podatku dochodowego prosimy także podać:
* oprócz nazwiska i imienia – także drugie imię,
* imiona rodziców,
* datę i miejce urodzenia,
* numer identyfikacyjny PESEL,
* miejsce zameldowania,
* adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora.

Tekst artykułu prosimy dostarczyć w formie maszynopisu lub wydruku komputerowego, pisanego jednostronnie, z podwójnym odstępem (30 wierszy na stronie i 60 znaków w wierszu).

Rysunki powinny być czytelne i mogą być wykonane ołówkiem.

Autor opublikowanego artykułu otrzymuje honorarium i bezpłatny egzemplarz okazowy.

Materiałów nie zamówionych redakcja nie zwraca.

Budzenie świadomości środowiska

dokończenie ze s. 1

Rzecznika Praw Obywatelskich. Ale zmiana nie jest sprawą miesiąca czy dwóch, lecz co najmniej roku. A w ciągu roku wiele rzeczy może się zdarzyć, szkoda więc naszego czasu i energii na tę walkę. Uznaliśmy, że znacznie ważniejsza jest sprawa funkcjonowania ustawy. Obecnie podjęliśmy stałą współpracę z BSA (*Business Software Aliance*), czyli międzynarodową organizacją zajmującą się w 50 krajach – w imieniu wielkich firm softwerowych – zwalczaniem piractwa komputerowego. Stowarzyszenie PRO będzie reprezentować w aliansie nasze firmy softwerowe, z BSA uzgodnimy sposób reprezentowania w Polsce firm zagranicznych, również od strony formalnej. Ustawa bowiem nie przewiduje możliwości zarządzania prawami autorskimi w Polsce przez podmioty zagraniczne. Nawiązaliśmy już współpracę z Biurem Ochrony Własności Intelektualnej, zostaną wykorzystane możliwości Fundacji Zwalczania Przestępczości Komputerowej, nawiążemy współpracę z MSW, policją, prokuraturą. Na pewno BSA pomoże nam w transferze technologii ścigania przestępców. Specjaliści z tej organizacji podejmą szkolenia polskich inspektorów specjalizujących się w zwalczaniu piractwa komputerowego. Ci zaś będą szkolić następnych. Niestety, w ustawie nie zostały uwzględnione nasze propozycje dotyczące przepisów wykonawczych. Ważna jest jednak ich interpretacja. Wprawdzie nic one nie mówią np. o prawie do wejścia do instytucji i przeprowadzaniu inwentaryzacji zawartości dysków twardych, ale prof. Jan Błeszyński, wybitny znawca prawa autorskiego uważa, że możliwa jest interpretacja przepisów ustawy pozwalająca na taką akcję. Mamy nadzieję, że pomogą nam biura adwokackie i mass media.

– Czyli czekają nas wkrótce procesy sądowe o naruszenie praw twórców oprogramowania?

– Jak wynika z praktyki BSA nie musi do tego dojść w skali masowej. BSA np. w USA zawiadamia kierownictwo określonej firmy, że podejrzewa ją o korzystanie z nielegalnego oprogramowania i pyta się czy ma wejść do niej inspektor – a prawo na to zezwala – by sprawdzić zawartość dysków twardych. Na ogół firma decyduje się na podjęcie negocjacji, uiszczenie należnych opłat i zapłacenie pewnej kwoty na rzecz BSA, która przeznacza ją na walkę z piractwem. Walczy się w zasadzie z firmami, a nie z prywatnymi użytkownikami. Metody stosowane przez BSA zdają egzamin, szczególnie w USA; obecnie nielegalne oprogramowanie stanowi tam ok. 35%, podczas gdy w Polsce ocenia się je na 94%. Myślę, że u nas zaostrzenie walki z piractwem komputerowym stanie się widoczne już w II połowie br.

– W imieniu redakcji i Czytelników gratuluję Panu nagrody i życzę powodzenia w budowaniu świadomości, że programy komputerowe są też wartością intelektualną zasługującą na ochronę.

GRZEGORZ CZAJKOWSKI
KRZYSZTOF ZIELIŃSKI
Katedra Informatyki
Akademia Górniczo-Hutnicza
Kraków

PCN
system rozproszonego przetwarzania

PCN[1] (*Program Composition Notation*) jest jednocześnie językiem i systemem programowania. Został opracowany przez zespoły z Argonne National Laboratory i California Institute of Technology. Jednym z głównych założeń projektu było [1, 2] zapewnienie wygodnej i prostej notacji, pozwalającej na myślenie (i programowanie) w kategoriach problemu, a nie języka programowania. Ponadto, istniejący już kod sekwencyjny powinien być wykorzystywany z jak najmniejszymi zmianami. Przez zapewnienie odpowiedniej komunikacji między procedurami napisanymi w C można niewielkim kosztem otrzymać rozproszoną, skalowalną, efektywnie wykonywaną aplikację, a dokonane wcześniej inwestycje nie muszą okazać się bezwartościowymi. Z doświadczenia autorów wynika, że PCN spełnia podane wyżej założenia.

Celem artykułu jest ogólne zapoznanie czytelnika z językiem PCN oraz bardziej szczegółowe przedstawienie wybranych aspektów programowania współbieżnego i rozproszonego. Dzięki prostej notacji prezentowany system umożliwia szybkie projektowanie i implementację programów równoległych. W języku PCN połączono zalety języków imperatywnych i deklaratywnych co jednak nie powoduje, że poznawanie systemu jest długie, a programowanie trudne.

Opis języka

Składnia PCN jest bardzo podobna do składni języka C; oprócz tego semantyka niektórych konstrukcji językowych jest zapożyczona z języków współbieżnego programowania w logice (warto tu wspomnieć, że uczestniczący w pracach nad PCN Steve Taylor i Ian Foster są autorami języka Strand'88). Ponadto przed kompilacją program jest przetwarzany przez preprocesor języka C, tak więc można korzystać np. z kompilacji warunkowej, czy stosować makrodefinicje. Wspomniane podobieństwa, jakkolwiek pomocne przy poznawaniu systemu, mogą być przyczyną wielu trudnych do wykrycia błędów, ponieważ programista, zasugerowany konstrukcją znaną z języka C (Prologu) jest skłonny traktować PCN jako rozszerzony C (Prolog). Dlatego też na końcu tej części artykułu znajdzie się kilka uwag, mogących zaoszczędzić sporo czasu i nerwów osobom poznającym PCN, a mającym utrwalone nawyki z innych języków programowania.

Typy danych

W obecnej wersji systemu dostępne są trzy proste typy danych: *char, int, double*. Możliwe jest tworzenie jednowymiarowych tablic danych tych typów; napisy są reprezentowane tak samo jak w C. Zbiór dostępnych operatorów jest w stosunku do C także bardzo skromny – można korzystać tylko z dodawania, odejmowania, mnożenia, dzielenia i obliczania reszty z dzielenia.

Złożone struktury danych są reprezentowane za pomocą krotek, np.: {1, *"napis"*, {2}}, oraz list, np. [1|Tail]. Zmienne w programach PCN można podzielić na dwie kategorie:
- zmienne, którym wartość można przypisywać wielokrotnie (ang. *mutable variables*) – nie mogą być one listami ani krotkami i muszą być zadeklarowane, np. *int var, char string* [16]. Do przypisywania wartości służy operator ":=";
- zmienne, którym wartość można przypisać dokładnie raz (ang. *definitional variables* – dalej użyto określenia *zmienne definicyjne*). Spełniają one nie tylko główną funkcję zmiennych (przechowywanie tymczasowych wartości obliczeń), lecz także służą do synchronizacji i komunikacji procesów; Zmienne tego typu nie są deklarowane. Do przypisania wartości służy operator "=".

Procedury i funkcje

Zapis procedury jest podobny do notacji języka C:
nazwa_procedury (*lista_parametrów*)
deklaracje
kompozycja

Sekcje *lista_parametrów* oraz *deklaracje* mogą być puste.

Deklaracja funkcji jest identyczna z deklaracją procedury poprzedzonej słowem *function*. Za pomocą instrukcji *return* (*ret_val*) funkcja może zwracać obliczoną wartość.

Ogólna postać kompozycji jest następująca:

{*op blok*1,...,*blokN*}

gdzie *op* jest jednym z trzech operatorów, oznaczających:
„?" – wybór; następujące po nim bloki są implikacjami – na podstawie wyników testów do dalszego wykonywania jest wybierana jedna z nich (w dalszej części artykułu omówiono zastosowanie operatora wyboru do synchronizacji obliczeń równoległych);
„;" – sekwencyjne obliczenia. Występujące po nim bloki są wykonywane sekwencyjnie;
„||" – równoległe obliczenia. Występujące po nim bloki są wykonywane równolegle.

Blok może być operacją przypisania, wywołaniem procedury, kompozycją lub *implikacją*. Implikacja to proste testy (sprawdzenie wartości zmiennej, jej typu, porównania arytmetyczne)

[1] System PCN można otrzymać w sieci Internet pod adresem info.mcs.anl.gov.

lub dopasowania złożonych struktur danych, po których następuje znak implikacji „–>" i blok. Jeżeli test sprawdza postać krotki lub listy (np. czy lista jest pusta czy też nie), używany jest operator „?=", np.:

List?=[ ] – error–msg („lista pusta")

Testy wchodzące w skład implikacji są ewaluowane od lewej do prawej, natomiast sekcje testów wszystkich implikacji są ewaluowane współbieżnie. Jeżeli nie są dostępne wystarczające dane do wykonania testu (np. zmienna definicyjna nie ma jeszcze nadanej wartości), testowanie jest wstrzymywane aż do chwili, gdy są dostępne wszystkie potrzebne informacje. Jeżeli prawdziwe są wszystkie testy kilku (w szczególnym przypadku – tylko jednej) implikacji, do dalszego wykonywania niedeterministycznie jest wybierany blok jednej z nich. W przypadku, gdy testy wszystkich implikacji dadzą wynik negatywny, następuje zakończenie wykonywania kompozycji wyboru i przejście do następnego bloku. Nie ma wymogu, aby testy co najmniej jednej implikacji były prawdziwe. Przykładem prostej procedury zawierającej implikacje jest wyznaczenie maksimum dwóch liczb:

```
max(a, b, c)
{?
   a > b -> c = a,
   default -> c = b
}
```

Warto podkreślić, że idea kompozycji jest kluczowym pomysłem dla systemu PCN. Za pomocą wyżej wymienionych trzech podstawowych operacji można łączyć ze sobą proste bloki, tworząc w ten sposób programy o dowolnym funkcjonalnym stopniu komplikacji. PCN wymusza czytelne i strukturalne programowanie (brak jest np. instrukcji *goto*). Ponadto, dzięki wygodnemu mechanizmowi korzystania z modułów jest bardzo ułatwione wielokrotne wykorzystywanie kodu oraz jego modularyzacja.

Model obliczeniowy

Model obliczeniowy PCN jest koncepcyjnie bardzo podobny do modelu współbieżnego programowania w logice. Na wykonywany program składa się zbiór lekkich procesów[2] komunikujących się ze sobą przez dzielone zmienne. Podstawowa metoda synchronizacji procesów jest oparta na dostępności danych – wykonywanie procesu czekającego na wartość zmiennej definicyjnej X, dzielonej z innym procesem, jest wstrzymywane do chwili, gdy X ma nadaną wartość. W ten sposób wykonanie procesów w PCN jest sterowane przepływem danych (ang. *data-flow*).

Uwagi dla programistów

Przed zaprezentowaniem kilku programów chcielibyśmy poczynić kilka uwag odnośnie tych aspektów systemu PCN, które mogą sprawić najwięcej kłopotów (i co najważniejsze, zabrać najwięcej czasu) osobom przyzwyczajonym do pisania w C lub w językach programowania w logice.

Uwagi te są następujące:

- Wszystkie parametry procedur i funkcji są przekazywane przez zmienną – nie jest tworzona kopia argumentu.
- Podobnie jak w Prologu, nie ma pojęcia adresu – przypisanie *str_war= "string"*, spowoduje, że wartością zmiennej *str_var* stanie się napis *"string"*, a nie jak w C, adres tego napisu.
- To, czy nazwa zmiennej rozpoczyna się od małej czy dużej litery nie ma żadnego związku z tym, czy wartość może być przypisywana jedno- czy wielokrotnie. O charakterze zmiennej decyduje tylko to, czy była zadeklarowana.
- Pojedyncze (w zakresie procedury) wystąpienie identyfikatora rozpoczynającego od małej litery, np. *var = counter*, nie jest (jak w Prologu) symbolem. Poprawne przypisanie powinno mieć postać *var = "counter"*.
- Nadmierna „ostrożność" w sensie unikania równoległości może prowadzić do trudnych do wykrycia błędów – niektóre obliczenia muszą być uruchomione jako równoległe, np. w programie 1 zastąpienie operatora " || " w procedurze *main* operatorem wykonania sekwencyjnego spowodowałoby zawieszenie się programu – wykonywanie procesów producenta i konsumenta musi przeplatać się w czasie.

```
main(argc, argv, exit_code)
{;
   {||
      producer(10, A, B),
      consumer(A, B)
   },
   exit_code = 0
}

producer(N, Out, Ack)
{||
   Out = [ N | Out1 ],
   {?
      Ack ?= [ _ | Ack1 ] -> producer(N - 1, Out1, Ack1),
      Ack ?= []-> {;}
   }
}

consumer(In, Ack)
{?
   In ?= [ 0 | _ ] -> Ack = [],
   In ?= [ H | In1 ], H > 0 ->
      {||
         Ack = [ _ | Ack1 ],
         consumer(In1, Ack1)
      }
}
```

Wydruk 1. Producent – konsument

- PCN nie jest językiem symbolicznym, m.in. operatory nie są funktorami – wyrażenia z funktorami są ewaluowane. Np. próba reprezentowania listy różnicowej w znany z Prologu sposób *Front-Back* jest błędem, podobnie jak 1 + 2 ma zawsze wartość 3 i nigdy nie może być interpretowana jako term "+" (1, 2).

Programowanie

Po przedstawieniu krótkiego opisu PCN chcielibyśmy zaprezentować kilka programów. Ilustrują one najważniejsze elementy języka i techniki programowania.

Komunikacja między procesami

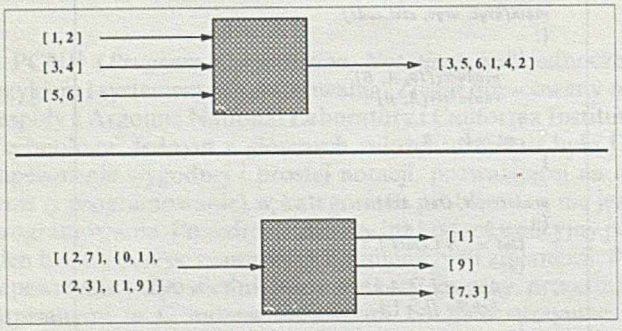
Jedyną metodą komunikacji między procesami programu napisanego w PCN jest dzielenie przez nie zmiennych definicyjnych[3]. Można w ten sposób przekazać jedną wartość lub dowolnie długą listę (twórcy systemu używają nazwy *strumień*) danych dowolnego typu. Korzystanie ze strumieni jest analogiczne do używania list w językach współbieżnego programowania w logice. Proces zawierający zmienną definicyjną X, dzieloną z innym procesem, chcąc przekazać daną H wykonuje przypisanie $X = [H|Xs]$. Zmienna Xs może być ponownie

[2] Nie są to lekkie procesy (wątki) w sensie systemu operacyjnego.

[3] Oczywiście, programując hybrydowo i łącząc C z PCN możemy wykorzystać wszystkie sposoby komunikacji między procesami, które oferuje system operacyjny danego komputera.

wykorzystana jako strumień. Strumień można zamknąć przypisując mu wartość listy pustej.

Opisana prosta komunikacja przez strumienie jest typowo jednokierunkowa. Pewnym rozwinięciem jest technika *incomplete messages* – przesyłana jest krotka, której jedno pole jest niezwiązaną zmienną definicyjną. Proces odbierający może w ten sposób przekazać wyniki swoich obliczeń procesowi, który zainicjował komunikację. W ten sposób można bardzo prosto zaprogramować obliczenia oparte na schemacie *master-worker*. Tworząc bardziej skomplikowaną i „demokratyczną" strukturę procesów, w której każdy może rozpocząć przesyłanie informacji do innych, trzeba jednak użyć więcej niż jednego strumienia. Przykładem może być program producent-konsument, przedstawiony na wydruku 1, gdzie dwa procesy komunikują się za pomocą dwóch strumieni.



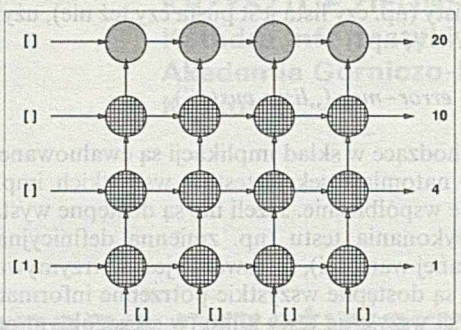
Wydruk 2. Przedstawienie działania procesu *merger* (u góry) i *distribute* (u dołu)

W tworzeniu bardziej skomplikowanych połączeń bardzo pomocne są predefiniowane procesy [4] *merger* (*In, Out*) oraz distribute (*N, In*) – wydruk 2. Pierwszy z nich pozwala na łączenie wielu strumieni w jeden i tym samym umożliwia komunikację typu *many-to-one*. Jeżeli strumień *In* ma postać [{"*merge*", *NewIn*} | *In*1], strumień *NewIn* jest łączony ze strumieniem *In*1. Zachowany jest wtedy wzajemny porządek elementów strumienia składowego w strumieniu wynikowym. Nie ma ograniczeń w zakresie liczby łączonych strumieni.

Proces *distribute* (*N, In*) ułatwia komunikację typu *one-to-many*. *N* określa maksymalną liczbę strumieni, do których są przesyłane dane. Elementy strumienia *In* mogą mieć jedną z następujących postaci:
{"*attach*", *N, S, Done*}, gdzie *N* jest liczbą naturalną, *S* jest strumieniem, natomiast *Done* jest zmienną definicyjną służącą do synchronizacji. Strumień *S* zostanie zarejestrowany pod numerem *N* w zbiorze strumieni, do których są wysyłane informacje.
{*N, Msg*} – dana *Msg* będzie przesłana do strumienia zarejestrowanego z numerem *N*.
{„*all*", *Msg*} – dana *MSG* zostanie przekazana do wszystkich strumieni wyjściowych.

Innym elementem języka ułatwiającym komunikację między procesami jest tablica portów (ang. *port array*). Deklarowana jako *port Name* [*Len*], jest listą strumieni, która może być indeksowana.

Przykładem użycia procesu *merger* oraz tablic portów jest program zilustrowany na wydruku 3, a przedstawiony na wydruku 4. Rozwiązuje on następujący problem: na ile sposobów można dojść z lewego dolnego wierzchołka kwadratowej siatki do dowolnego jej węzła w prawym boku, przy założeniu, że można poruszać się tylko w prawo i w górę. Proces *grid* umieszcza kolejno kolumny siatki i zapewnia komunikację między sąsiednimi procesami, przez użycie tablic portów. Z kolei proces *column* tworzy procesy wchodzące w skład danej kolumny.



Wydruk 3. Schematyczne przedstawienie problemu z sekcji 3.1.
Każde koło, to element siatki (proces, węzeł). Na początku lewy dolny proces otrzymuje jedną informację. Strumienie dolny i lewy są łączone i przekazywane do góry i w prawo (informacja zostaje powielana). Węzły siatki tworzące górny bok przekazują otrzymane informacje tylko w prawo. Zadaniem programu jest wyznaczenie, ile wiadomości otrzyma każdy węzeł leżący na prawym boku siatki.

```
go(N, WhichOutput, Out)
port I[N], O[N];
{;
   i over 0 .. N - 2 :: I[i] = [],
   I[N - 1] = [ 1 ],
   grid(N, N, I, O),
   Out = length(O[WhichOutput])
}

grid(N, M, InP, OutP)
port InP[], MP[M], OutP[];
{?
   N == 1 -> column(M, [], InP, OutP),
   default -> {||
      column(M, [], InP, MP),
      grid(N - 1, M, MP, OutP)
   }
}

column(M, FromDown, FromLeftP, ToRightP)
port FromLeftP[], ToRightP[];
{?
   M == 1 -> w(FromDown, _, FromLeftP[0], ToRightP[0], "yes"),
   default -> {||
      w(FromDown, ToUp, FromLeftP[M - 1], ToRightP[M - 1], "no"),
      column(M - 1, ToUp, FromLeftP, ToRightP)
   }
}

w(FD, TU, FL, TR, Border)
{||
   sys:merger([ { "merge", FD }, { "merge", FL } ], Is),
   {?
      Border == "yes" -> TR = Is,
      default -> {|| TR = Is, TU = Is }
   }
}
```

Wydruk 4. Rozwiązanie opisanego problemu kombinatorycznego

Rekurencja oraz iteracja

Wymienione wcześniej operatory wyboru oraz wykonywania sekwencyjnego i równoległego są podstawowymi elementami języka.

Istnieje tylko jedna iteracyjna instrukcja w języku:

{*op index over low .. high :: block*}

w wyniku której *blok block* zostanie wykonany dla każdej liczby całkowitej *index*, takiej, że $low \leqslant index \leqslant high$. Przy programowaniu z wykorzystaniem tablic użycie tej instrukcji jest znacznie wygodniejsze niż programowanie z wykorzystaniem rekurencji.

Przykładem może być program obliczania iloczynu skalarnego (wydruk 5). Wersja napisana z użyciem instrukcji iteracyjnej jest bardzo podobna do programu napisanego w językach imperatywnych, natomiast procedura wykorzystująca rekurencję przypomina programowanie w logice.

```
a)
inner_i(len, A, B, C)
int A[], B[], sum;
{;
   sum := 0,
   {; i over 0 .. len - 1 :: sum := sum + A[i] * B[i] },
   C = sum
}

b)
inner_r(A, B, C)
{?
   A ?= [ HA | A1 ], B ?= [ HB | B1 ] ->
      {;
         inner_r(A1, B1, C1),
         C = C1 + HA * HB
      },
   A ?= [] -> C = 0
}
```

Wydruk 5. Obliczanie iloczynu skalarnego:
a) procedura wykorzystująca tablice, iterację oraz zmienne, którym wartość można przypisywać wielokrotnie
b) procedura wykorzystująca listy, rekurencję oraz zmienne definicyjne.

Język PCN, nie oferując żadnych innych instrukcji iteracyjnych, zmusza programistę do częstego stosowania rekurencji, która staje się dzięki temu podstawową metodą przekazywania sterowania. Zaletą takiego podejścia jest czytelność i przejrzystość programów (w większości przypadków). Ponadto stosowanie rekurencji nie jest w PCN tak kosztowne czasowo, jak np. w C. Wynika to z faktu, że obliczenia w PCN są wykonywane przez zbiór współbieżnie wykonywanych lekkich procesów, a nie przez sekwencyjnie wykonywane, obsługiwane przez stos procedury. Dla czasu wykonywania programu w PCN nie jest znacząco istotne, czy procesy powstaną w wyniku użycia rekurencji, czy iteracji – natomiast w C zastąpienie np. pętli *while* rekurencyjnym wywołaniem procedury z reguły spowalnia program.

Obliczenia rozproszone i topologie wirtualne[4)]

Każdy program napisany w PCN może wykonywać się zarówno na jednym, jak i na wielu komputerach. Według naszych testów nie ma żadnych problemów z poprawnym wykonaniem na wielu stacjach roboczych programu uruchomionego wcześniej na jednym komputerze. Dużą pomocą we wstępnej fazie projektowania rozproszonej aplikacji jest pojęcie *topologii wirtualnych*. Topologia wirtualna jest zdefiniowaną przez programistę lub predefiniowaną abstrakcją struktury połączeń między maszynami wirtualnymi. Topologia wirtualna może liczyć więcej maszyn wirtualnych, niż jest dostępnych komputerów w sieci. Topologie wirtualne mogą być zmieniane dynamicznie w trakcie wykonywania programu. Ponadto, każda topologia wirtualna może być podstawową topologią dla innej, bardziej atrakcyjnej topologii. Można w ten sposób zbudować hierarchię topologii wirtualnych.

Informacji na temat aktualnej wirtualnej topologii dostarczają następujące funkcje:

- *topollgy*( ) – zwraca krotkę z informacją o topologii, np. {*"array"*, 8};
- *nodes*( ) – zwraca liczbę (wirtualnych) procesorów w topologii;
- location ( ) – zwraca informaję o (wirtualnym) procesorze, na którym dany proces jest wykonywany.

Funkcją definiującą topologię wirtualną może być dowolna funkcja zwracająca krotkę postaci {*type, size, map_fn*}, gdzie *type* jest daną informującą o typie topologii, np. {*"star"*, 7} (tę daną zwraca następnie wywołanie funkcji *topology*( )), *size* to liczba żądanych (wirtualnych) procesorów, natomiast *map-fn* jest funkcją przyporządkowującą procesowi procesor. Np. topologię typu *ring* można zadać za pomocą funkcji:

function node (*i*) {; *return* (*i* % *nodes* ( ))}

Jeżeli funkcja definiująca topologię *ring* nazywa się *ring_in_array*, to wywołanie:

proc(../*args*/..) *in ring_in_array*(../*args*/..)

spowoduje, że topologią dla procesów potomnych procesu *proc* jest *ring* (standardową, domyślną topologią jest *array*), natomiast wywołanie:

p(../*args*/...) ⓐ *node*(*N*)

umieści proces *p* na *N*-tym komputerze w zdefiniowanym pierścieniu. Wiele użytecznych funkcji i procedur pomocnych w tworzeniu i zarządzaniu topologiami wirtualnymi znajduje się w module *vts*, dostarczanym wraz z systemem PCN.

```
mapper(In)
{?
   In ?= [] -> {;},
   In ?= [ { "first", Pos } | In1 ] -> {|| Pos = 1, mapper(In1) },
   In ?= [ { "kids", ParentPos, Pos1, Node1, Pos2, Node2 } | In1 ] ->
      {? ParentPos > -1 ->
         {||
            Pos1 = ParentPos * 2,
            Pos2 = ParentPos * 2 + 1,
            Node1 = Pos1 % nodes(),
            Node2 = Pos2 % nodes(),
            mapper(In1)
         }
      }
}

qsort(In, Out, FirstPos, ToMapper)
{|| qs(In, Out, [], FirstPos, ToMapper ) }

qs(In, DFront, DTail, Pos, ToMapper )
{?
   In ?= [] -> {|| DFront = DTail, ToMapper = [] },
   In ?= [ H | T ] ->
      {;
         partition(H, T, Bigs, Smalls),
         ToMapper = [ { "kids", Pos, Pos1, Node1, Pos2, Node2 },
                      { "merge", ToMapper1 } | ToMapper2 ],
         qs(Smalls, DFront, [ H | DMiddle ], Pos1, ToMapper1)@vts:node(Node1),
         qs(Bigs, DMiddle, DTail, Pos2, ToMapper2 )@vts:node(Node2)
      }
}

partition(Pivot, In, B, S)
{?
   In ?= [] -> {|| B = [], S = [] },
   In ?= [ H | T ] ->
      {?
         Pivot > H -> {|| S = [ H | S1 ], partition(Pivot, T, B, S1) },
         default -> {|| B = [ H | B1 ], partition(Pivot, T, B1, S) }
      }
}
```

Wydruk 6. Rozmieszczenie procesów realizujących sortowanie szybkie (proces *mapper* zarządza lokalizacją)

[4)] Autorzy artykułu przetestowali przedstawiane programy na sieci stacji roboczych SUN4/SPARC2, stąd też termin *obliczenia rozproszone*. System PCN jest jednak zaimplementowany nie tylko na sieci stacji roboczych – dostępne są także wersje na maszyny wieloprocesorowe.

Alternatywną możliwością dynamicznego sterowania rozmieszczeniem obliczeń jest zdefiniowanie procesu odpowiedzialnego za przydział zasobów. Przykład pokazuje definicję takiego zarządcy procesorów. Implementuje on topologię typu *binary tree* i odpowiada na następujące pytania:

- gdzie umieścić pierwszy proces?
- gdzie umieścić dwa procesy potomne, dysponując informacją o lokalizacji procesu pytającego?

Algorytm jest prosty – topologia drzewa binarnego jest implementowana na topologii tablicy. W podanym przykładzie proces *mapper* spełnia funkcję, którą mógłby zapewnić mechanizm topologii wirtualnych. Możliwości są tu jednak znacznie większe – można np. podejmować decyzje poprawiające *load-balancing* – można np. podejmować decyzje poprawiające *load-balancing* na podstawie przechowywalnych informacji o liczbie procesorów na danej maszynie, czy też wypisywać diagnostyczne i statystyczne komunikaty. Obliczeniami korzystającymi z usług procesu *mapper* są procesy sortujące listę metodą *quicksort* (szybkiego sortowania – rys. 6). Uogólnienie podanego przykładu można znaleźć w [4].

Metaprogramowanie

PCN umożliwia korzystanie z funkcji wyższego rzędu. Nazwa wywoływanej procedury i jej modułu może być zmienną, wyznaczaną w trakcie wykonywania programu. Wartością zmiennej jest napis oznaczający nazwę procedury lub modułu. Przy wywołaniu zmienna powinna znajdować się między odwróconymi apostrofami (rys. 7).

```
map(op, List, Result)
{?
   List ?= [] -> Result = [],
   List ?= [ H | T ] ->
      {||
         'op'(H, Val),
         Result = [ Val | Result1],
         map(op, T, Result1)
      }
}

foldl(op, List, SoFar, Result)
{?
   List ?= [] -> Result = SoFar,
   List ?= [ H | T ] ->
      {||
         'op'(SoFar, H, Val),
         foldl(op, T, Val, Result)
      }
}
```

Wydruk 7. Przykłady znanych funkcji wyższego rzędu

Programowanie z użyciem powstałych w ten sposób *metawywołań* ma wiele zastosowań, od znanych w językach funkcjonalnych funkcji wyższego rzędu, takich jak *map* czy *foldl*, aż do możliwości wielokrotnego wykorzystywania tej samej struktury połączeń między procesami dla różnych obliczeń. Program z rys. 8 jest przykładem wykorzystania opisanych możliwości. Wywołanie procedury *make_loop* (*ProcName, LoopingList*), gdzie *ProcName* jest nazwą procedury, a *LoopingList* ma postać $[[a1, s1, e1], .., [aN, sN, eN]]$, jest równoważne następującemu zapisowi w pseudokodzie:

$for(i1 = a1; i1 \leqslant e1; i1 += s1)$

. . . . . .

$for(iN = aN; \quad iN \leqslant eN; \quad iN += sN)$

ProcName $([i1, .., iN])$

Bez trudu można uogólnić procedurę *make_loops* na np. różne typy parametrów procedury *ProcName*. Ponieważ jedyną

```
make_loop(ProcName, Formals) {; loop(ProcName, Formals, []) }

loop(ProcName, ParamList, Actuals)
{?
   ParamList ?= [] ->
      {;
         reverse(Actuals, RvActuals),
         'ProcName'(RvActuals)
      },
   ParamList ?= [ [ Start, Step, End ] | T ] ->
      {?
         Start > End -> {;},
         Start <= End ->
            {;
               loop(ProcName, T, [ Start | Actuals ]),
               loop(ProcName, [ [ Start + Step, Step, End ] | T ], Actuals)
            }
      }
}
```

Wydruk 8. Wykonywanie pętli przy użyciu metaprogramowania

możliwością iterowania jest omówiona wcześniej instrukcja *over*, z licznikiem całkowitym oraz z krokiem zawsze równym 1, do programowania bardziej ogólnych pętli konieczne jest wykorzystanie rekurencji. Tworzenie w ten sposób, tzn. z użyciem rekurencji (głęboko) zagnieżdżonych pętli zmniejsza czytelność programu, jest czasochłonne i podatne na błędy; jest to także przykład zagadnienia, w którym użycie rekurencji zmniejsza czytelność rozwiązania. Dlatego też możliwość tworzenia procedur „ogólnego przeznaczenia", takich jak na rys. 8, staje się atrakcyjna. Po przetestowaniu i udokumentowaniu *make_loops* może być wykorzystywana wielokrotnie.

Ostatnim prezentowanym przykładem jest bardzo prosta *rama komunikacyjna*, stanowiąca ogólny schemat obliczeń w topologii typu *ring* (rys. 9). Procedura *ProcName* oczekuje dwóch argumentów – strumienia wejściowego i wyjściowego.

```
ring(ProcName, S)
port S[nodes()];
{|| i over 0 .. nodes() - 1 :: 'ProcName'(S[(i + 1) % nodes()], S[i]) @ vts:node(i)
}
```

Wydruk 9. Procedura rozsyłająca procesy *ProcName* w topologii typu pierścień

* * *

PCN wypełnia pewną lukę w zakresie środowisk do tworzenia niezależnych sprzętowo aplikacji rozproszonych i równoległych. Programy są przenośne, istnieje możliwość efektywnego wykorzystania istniejącego kodu sekwencyjnego. Język udostępnia deklaratywne i imperatywne mechanizmy programowania. Wygodna komunikacja między procesami oraz możliwość metaprogramowania są także zaletami systemu PCN. Mechanizm topologii wirtualnych pozwala na testowanie aplikacji w różnych topologiach oraz powoduje, że programy napisane w PCN są skalowalne.

PCN jest istotnym krokiem naprzód w zakresie programowania systemów rozproszonych i równoległych. Należy spodziewać się dalszych interesujących wyników badań w tym kierunku.

LITERATURA

[1] Cunningham H.C.: Notes on Concurrent Programming with PCN. 1992

[2] Foster I., Olson R., Tuecke S.: Productiwe Parallel Programming: The PCN Approach. Scientific Programming, Vol. 1 pp. 51–66, 1991

[3] Foster I., Tuecke S.: Parallel Programming with PCN. Technical report, Argonne, Argonne National Laboratories, 1992

[4] Gąsiorowska A., Laurentowski A.: Ramy i ich zastosowanie do enkapsulacji interakcji między procesami w systemach rozproszonych. Praca magisterska, Akademia Górniczo-Hutnicza im. St. Staszica w Krakowie, lipiec 1993.

Progress 4GL/RDBMS

System Progress przeznaczony jest do tworzenia i eksploatacji aplikacji obsługujących relacyjne bazy danych. Progress zawiera zintegrowane środowisko pracy dla projektanta i programisty oraz system zarządzania relacyjną bazą danych, pod kontrolą którego wykonywane są aplikacje.

Progress RDBMS *(Relational Database Management System)* jest bardzo wydajnym systemem zarządzania bazą danych dla rzeczywistych aplikacji. Jest on przy tym systemem skalowalnym, co oznacza, że efektywność przetwarzania danych wzrasta proporcjonalnie do zasobów systemu.

Progress jest systemem otwartym, co oznacza m.in., że możliwa jest współpraca z innym oprogramowaniem oraz tworzenie i użytkowanie aplikacji progressowych na różnych platformach sprzętowych, a także dowolne rozbudowywanie czy modyfikacja posiadanej konfiguracji bez potrzeby dokonywania zmian w aplikacji.

Progress zawiera narzędzia wspomagające pracę programisty i znacznie przyspieszające proces tworzenia aplikacji. Sam język programowania Progress 4GL jest językiem wysokiego poziomu, o czytelnej i zwięzłej składni, dzięki czemu łatwe jest zarówno uczenie się go, jak i stosowanie.

Centralnym elementem systemu jest rozbudowany słownik bazy danych (Data Dictionary), który przechowuje szczegółowe informacje o strukturze bazy, powiązaniach relacyjnych pomiędzy poszczególnymi plikami, strukturze indeksów. W Słowniku zawarte są także reguły określające więzy integralności bazy i reguły walidacyjne na poziomie pól w rekordzie. Również na poziomie słownika definiowane są triggery, czyli procedury uruchamiające się samoczynnie w przypadku zajścia określonego zdarzenia związanego z bazą danych. Słownik zawiera też pewne informacje pomocnicze takie, jak etykiety i formaty używane przy wyświetlaniu danych, teksty podpowiedzi pojawiające się na ekranie przy wprowadzaniu lub modyfikowaniu danych czy wartości domyślne dla poszczególnych pól. Ponadto, jedną z opcji dostępnych przy pracy ze Słownikiem Bazy Danych jest możliwość drukowania raportów obrazujących pełną strukturę bazy; raporty te mogą stanowić dokumentację techniczną projektu, co usprawnia pracę zespołu, a także ułatwia późniejsze utrzymanie aplikacji.

Progress umożliwia korzystanie z dwukierunkowych pomostów *(bridges)* do programów CASE wysokiego poziomu. Po wykonaniu projektu aplikacji przy pomocy programu CASE następnym krokiem jest generacja (przy użyciu pomostu) schematu bazy danych Progress. Pomosty umożliwiają także tzw. *reverse engineering*, czyli wczytanie gotowego (stworzonego ręcznie) schematu bazy danych do narzędzia CASE w celu weryfikacji lub też zmodyfikowania tego schematu.

Język Progress 4GL jest językiem kompletnym, co oznacza, że cała aplikacja może być napisana w Progressie bez konieczności odwoływania się do języków niższego poziomu. Progress umożliwia programowanie sterowane zdarzeniami *(ang. event driven programming)*, co daje w efekcie aplikację o bardziej płaskiej strukturze, a więc bardziej elastyczną i przyjazną dla użytkownika. Progress 4GL zawiera wbudowany język SQL 89 Level 2.

Zintegrowane środowisko pracy dla programisty zawiera następujące elementy: Słownik Bazy Danych, edytor procedur, kompilator, generator graficznego interfejsu użytkownika oraz debugger.

Progress daje użytkownikowi pełną gwarancję całkowitego zabezpieczenia przechowywanych danych przed zniszczeniem lub utratą spójności czy też przed niepowołanym dostępem. Możliwa jest pełna i przyrostowa archiwizacja danych *(backup)*, a także archiwizacja on-line. Blokowanie zasobów w systemach wielodostępnych odbywa się na poziomie rekordu.

Progress jest ukierunkowany na przetwarzanie transakcyjne, co zapewnia utrzymanie spójności logicznej bazy danych. Poprawność wykonania transakcji rozproszonych jest zagwarantowana poprzez zastosowanie dwufazowego protokołu potwierdzeń *(two-phase commit)*.

Dodatkowe możliwości systemu Progress to multimedialność, indeksowanie słów w polach tekstowych i możliwość tworzenia aplikacji wielojęzycznych.

KONFERENCJA KLUBU UŻYTKOWNIKÓW PROGRESSA

W dniach 15-17 czerwca odbędzie się w Cetniewie III Ogólnopolska Konferencja Polskiego Klubu Użytkowników Systemu Progress. Organizatorzy przewidują, że w Konferencji weźmie udział ok. 150 uczestników, zarówno aktualnych użytkowników Progressa, jak i przedstawicieli firm zainteresowanych tym systemem. Celem konferencji jest zarówno pogłębienie znajomości Progressa i podniesienie poziomu wiedzy informatycznej uczestników, jak i integracja środowiska użytkowników. Wszelkie informacje o Konferencji można uzyskać w Zespole Progress CSBI, ul. Sapieżyńska 3, 02-119 Warszawa, tel. 31-37-47, fax 635-65-58.

0/2/94

MAREK MIŁOSZ
ROBERT PRANAGAL
Katedra informatyki
Politechnika Lubelska

Budowa prototypu systemu eksportowego

Próby tworzenia systemów ekspertowych sięgają lat sześćdziesiątych, ale dopiero ostatnio systemy te stały się bardzo modne. Sztuczna inteligencja, inżynieria wiedzy, analiza i rozumienie obrazów i mowy, przetwarzanie języka naturalnego, szkieletowe systemy ekspertowe to pojęcia, które dzięki rozwojowi techniki komputerowej i oprogramowania opuściły laboratoria naukowe i weszły do codziennego języka zastosowań informatyki. Równolegle z rozwojem i komercjalizacją systemów ekspertowych są opracowywane różne techniki i metodologie ich tworzenia.

Mimo, że systemów ekspertowych powstaje (i użytkuje się) coraz więcej, to dokładne informacje na ich temat są przeważnie ścisłą tajemnicą firm [9]. Jeszcze mniej znane są sposoby budowy konkretnych systemów. Jednym z zastosowań sztucznej inteligencji jest wspomaganie analizy stanu skomplikowanych systemów (np. sieci elektroenergetycznych).

Rosnące zapotrzebowanie na energię elektryczną oraz coraz większe wymagania odnośnie jej jakości pociągają za sobą ciągłą rozbudowę systemów energetycznych. Systemy, zapewniające dużą pewność oraz elastyczność zasilania są coraz bardziej skomplikowane. Elastyczność pociąga za sobą konieczność ciągłego przetwarzania ogromnej ilości informacji. Ilość ta gwałtownie rośnie w przypadku wystąpienia zakłóceń w systemie energetycznym (np. podczas burzy). Jednocześnie czas na podjęcie decyzji jest bardzo krótki. Szybkie odizolowanie uszkodzenia często chroni cały system przed destrukcją. Ponowna konfiguracja systemu jest czasochłonna, kosztowna (koszt niedostarczonej energii) i wymaga pełnej koordynacji działań. Celem niedopuszczenia do takich sytuacji stosuje się skomplikowane układy automatyki i zabezpieczeń.

Systemy takie nie są jednak idealne i dlatego nie można wyeliminować roli operatora, który musi nadzorować pracę systemu oraz interweniować w sytuacjach wyjątkowych. Operatorzy nadzorujący duże systemy energetyczne powinni być fachowcami o najwyższych kwalifikacjach i wielkim doświadczeniu, co powoduje, że koszt ich wyszkolenia jest znaczny. Jednym ze sposobów rozwiązania tego problemu jest zastosowanie do wspomagania działań wspomnianych operatorów systemów ekspertowych (SE), zwłaszcza pracujących w czasie rzeczywistym [6]. Zakup i użytkowanie SE jest jednak znacznie tańsze i szybsze od wyszkolenia i opłacenia wysokokwalifikowanego eksperta. Systemy ekspertowe są również pomocne przy szkoleniu i podnoszeniu kwalifikacji operatorów, dzięki możliwości symulowania sytuacji wyjątkowych i niebezpiecznych. Ponadto stosując SE do obsługi systemu energetycznego obniża się wymagania w zakresie niezbędnego poziomu doświadczenia operatorów.

Dr inż. MAREK MIŁOSZ ukończył w 1982 r. studia wyższe, a w 1986 r. studia doktoranckie na Wydziale Automatycznych Systemów Zarządzania Moskiewskiego Instytutu Inżynieryjno-Budowlanego w Moskwie (Rosja). Od 1982 roku jest pracownikiem naukowo-dydaktycznym Katedry Informatyki Politechniki Lubelskiej. Zajmuje się problemami projektowania, programowania i wdrażania systemów informatycznych, symulacji procesów i ogólnie pojętymi inżynierskimi zastosowaniami informatyki. Jest autorem lub współautorem ponad 50 publikacji naukowych i 15 projektów naukowo-badawczych i wdrożeniowych.

Obiekt: rozdzielnia wysokiego napięcia

Próbę budowy prototypu SE podjęto dla fragmentu systemu elektroenergetycznego zakładów WSK PZL Świdnik: rozdzielni wysokiego napięcia RG-2. Rozdzielnia ta jest zasilana z dwóch niezależnych linii o napięciu znamionowym 110 kV i pracuje w układzie *H5* z szynami w układzie dwusekcyjnym dwusystemowym.

Na każdą sekcję przypada po 16 pól różnego typu. Każde z tych pól ma obwody sterowania, zabezpieczeń, sygnalizacji i blokad. Są to wprawdzie obwody odrębne, ale między nimi istnieją powiązania, powodujące dodatkowe komplikowanie analizy stanów rozdzielni.

Metodologia budowy systemu ekspertowego

Podstawową rolę w tworzeniu systemów ekspertowych stanowi inżynier wiedzy [5, 9]. Powinien on znać metody gromadzenia i formalizacji wiedzy, języki programowania wysokiego poziomu (LISP, PROLOG) lub szkieletowe systemy ekspertowe (ang. *Expert System Shell*) oraz metody testowania SE. Inżynier wiedzy nadzoruje cały proces tworzenia systemu ekspertowego. Poszczególne etapy tego procesu zostaną omówione w dalszej części artykułu. W praktyce okazuje się, że granice te zacierają się, a czasem między poszczególnymi etapami występują sprzężenia zwrotne. Metodologia tworzenia SE przypomina typowy proces tworzenia oprogramowania [5] i składa się z następujących etapów:

1) zdefiniowania problemu,
2) gromadzenia wiedzy,
3) formalizacji wiedzy,
4) implementacji wiedzy,
4) testowania systemu.

Zdefiniowanie problemu

Pierwszą fazą tworzenia SE jest zdefiniowanie problemu tzn. określenie zadania jakie system ma rozwiązywać. Już na

początku tego etapu inżynier wiedzy powinien ustalić źródło wiedzy, z jakiego będzie korzystał. Najczęściej jest to ekspert z dziedziny, w której system ma działać. Sposób wyboru eksperta jest sprawą istotną, a jednocześnie trudną, ale dotyczy on raczej sfery stosunków międzyludzkich niż sfery merytorycznej.

Inżynier wiedzy może i powinien korzystać z publikacji jako podstawowych źródeł wiedzy. Po ustaleniu konkretnego źródła wiedzy (najczęściej po dokonaniu wyboru przez eksperta), inżynier wiedzy wspólnie z ekspertem dokonują szczegółowego określenia problemu.

Oczywiście nie wszystkie problemy są jednakowo podatne na rozwiązanie za pomocą systemu ekspertowego. Problem, dla którego ma być opracowany SE powinien cechować się [10]:
- możliwie wąską specjalizacją,
- precyzyjną definicją
- brakiem algorytmów rozwiązywania problemu,
- brakiem oparcia na wiedzy zdroworozsądkowej.

Stworzenie systemu dla rozwiązania konkretnego problemu nie zawsze jest opłacalne. Aby ocenić podatność problemu na rozwiązanie za pomocą SE należy rozpatrzeć [8]:
- definicję zadania,
- istniejące rozwiązania zadania,
- dostępność eksperta,
- zapotrzebowanie na konkretny system ekspertowy,
- wymagania sprzętowe systemu,
- czas i koszt budowy systemu.

Gromadzenie wiedzy

Proces gromadzenia (nabywania) wiedzy polega na zdefiniowaniu wiedzy, która posłuży systemowi ekspertowemu za podstawę do udzielenia odpowiedzi. W praktyce definiowanie może odbywać się w sposób [10]:
1) nieautomatyczny, przez wywiad z ekspertem (uporządkowany lub nieuporządkowany) oraz śledzenie działań eksperta (obserwacje, analizy protokołów, ksiąg itp.) lub kombinację tych metod;
2) półautomatyczny, ukierunkowany na wspomaganie eksperta oraz wspomaganie inżyniera wiedzy;
3) automatyczny, przez automatyczne tworzenie reguł oraz maszynowe uczenie się.

Gromadzenie wiedzy w sposób nieautomatyczny, a więc drogą jej pozyskania od eksperta, jest obecnie metodą najczęściej stosowaną. Wiedza powinna być w postaci łatwej do sformalizowania. W etapie tym odbywa się pełna współpraca eksperta, który odpowiada za treść, z inżynierem wiedzy, odpowiedzialnym za formę oraz zdefiniowanie reguł wnioskowania. Inżynier wiedzy może wykonać swoje zadanie albo tworząc system od podstaw (przy użyciu języków wysokiego poziomu, takich jak LISP czy PROLOG), albo korzystając ze wspomnianych na wstępie szkieletowych systemów ekspertowych, tj. systemów nie zawierających baz wiedzy. Metody te, zwane także „ręcznymi", są najbardziej czasochłonne, kosztowne, a często również niedokładne, co skłania do automatyzacji tego procesu.

Postępująca automatyzacja procesu gromadzenia wiedzy wykazuje tendencję do ograniczenia roli inżyniera wiedzy, a nawet jego eliminacji, natomiast ekspert jest tu nieodzowny. Oczywiście można sobie wyobrazić eliminację udziału eksperta w sytuacji, gdy podstawowym źródłem wiedzy dla inżyniera wiedzy będzie np. literatura. Metoda ta jest jednak czasochłonna i wymaga szczegółowej aktywności inżyniera wiedzy, który w zasadzie zmienia się wówczas w eksperta z danej dziedziny. W każdym przypadku jednak inżynier musi rozumieć istotę rozwiązywanego problemu. Pomaga to w ukierunkowaniu procesu gromadzenia wiedzy, podczas którego konieczne jest [7]:
- podzielenie problemu na podproblemy,
- określenie parametrów systemu,
- określenie zbioru rozwiązań,
- określenie informacji niezbędnych do rozwiązania problemu,
- określenie relacji między poszczególnymi wielkościami.

Jako produkt końcowy procesu gromadzenia wiedzy inżynier wiedzy otrzymuje opis wszystkich wielkości oraz ich wzajemne zależności w postaci grafów, tabel, diagramów itp.

Gromadzenie wiedzy z udziałem inżyniera wiedzy jest obecnie sposobem najbardziej rozpowszechnionym i najlepiej znanym. Współczesne trendy zmierzają jednak w kierunku jego wyeliminowania z procesu gromadzenia, formalizacji i implementacji wiedzy, tzn. zapewnienia bezpośredniego kontaktu eksperta z systemem. Niestety kontakt taki jest niemożliwy wtedy, gdy problem nie daje się zaimplementować w dostępnych szkieletowych systemach ekspertowych. W takiej sytuacji konieczne jest opracowanie systemu od podstaw.

Istnieje coraz więcej systemów szkieletowych, umożliwiających ekspertowi samodzielne wypełnianie bazy wiedzy oraz uruchamianie konkretnych systemów ekspertowych. Podstawową przeszkodą i trudnością jest tu stworzenie systemu komunikacji z użytkownikiem (ang. *user interface*), który pozwoliłby ekspertowi-nieinformatykowi wprowadzać wiedzę w łatwy i przejrzysty sposób.

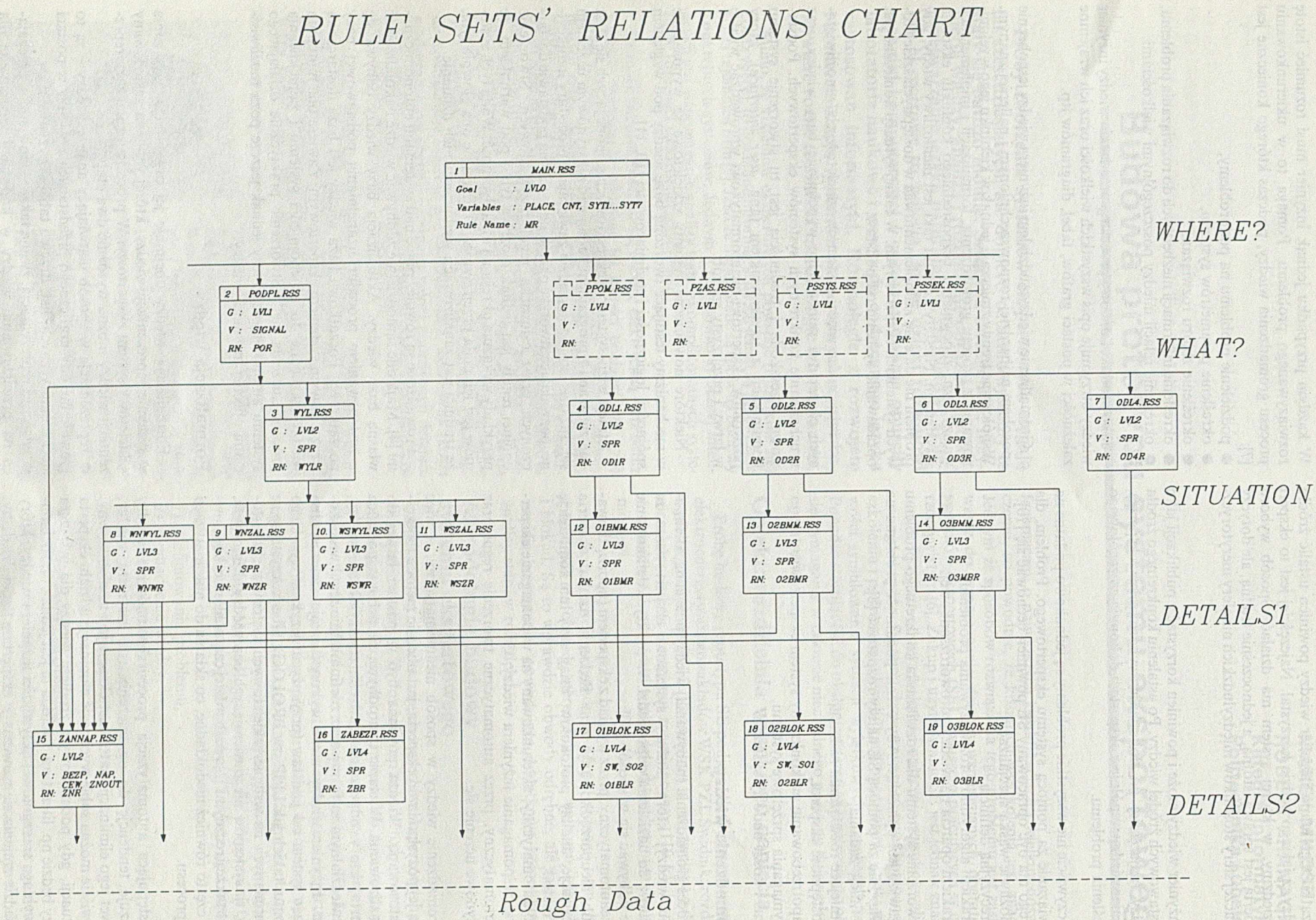
Niektóre nowoczesne systemy szkieletowe są wyposażone w mechanizmy testujące tworzoną bazę wiedzy pod względem spójności, logiczności i nadmiarowości [10, 11].

Istnieje wizja systemów ekspertowych samouczących się. Jest ona związana z programem badań nad maszynowym uczeniem się. Już dziś istnieją narzędzia zawierające procedury indukcyjne, pozwalające stworzyć określone reguły na podstawie dostarczonych przez człowieka reguł przykładowych. Stworzenie ogólnej reguły jest jednak możliwe tylko pod warunkiem, że przykłady wykazują określoną metodyczność. Wadą tych systemów jest ich mała przejrzystość. Objawia się ona głównie tym, że trudniej jest zanalizować i wytłumaczyć ich działanie.

Prowadzone są również prace nad systemami autonomicznymi [4]. Mają one samodzielnie zdobywać wiedzę i aktualizować własną bazę wiedzy. Aktualizacja bazy wiedzy odbywa się z zastosowaniem procedur uzupełniania podstawowej bazy początkowej. Początkowa baza wiedzy jest bazą dostarczoną wraz z nowym systemem ekspertowym. Zawiera ona minimum wiedzy niezbędne do uruchomienia systemu i zainicjowania procesu „uczenia się". Stworzenie praktycznie działającego systemu autonomicznego leży jednak jeszcze poza możliwościami obecnej inżynierii wiedzy.

Formalizacja wiedzy

Formalizacja wiedzy obejmuje jej organizację oraz zapis w systemie reprezentacji wiedzy, której wymaga zastosowany szkieletowy system ekspertowy. W procesie formalizacji reprezentacji wiedzy można wyróżnić dwa etapy:
- reprezentacja w sposób ułatwiający analizę wiedzy – są to głównie metody graficznego przedstawienia wiedzy w postaci sieci semantycznych, list, drzew i tabel decyzyjnych;
- właściwa reprezentacja, zwana także kodowaniem, polegająca na przedstawianiu wiedzy w konwencji zrozumiałej dla



Rys. 1. Proces budowy systemu ekspertowego

szkieletowego systemu ekspertowego. Czasem etap pierwszy jest pomijany.

Pomimo, że niektóre systemy szkieletowe umożliwiają wyeliminowanie z procesu formalizacji inżyniera wiedzy, to jednak powinien on dokonać wyboru samego systemu. O wyborze decyduje w głównej mierze sposób reprezentacji wiedzy. Powinien on być taki, aby odpowiedzi systemu ekspertowego były możliwie zbliżone do odpowiedzi udzielanych przez eksperta. Wskazane jest użycie kilku metod reprezentacji wiedzy i przetestowanie, która z nich jest najlepiej dostosowana do konkretnego problemu.

Reprezentacja wiedzy jest zdecydowanie najtrudniejszą częścią procesu tworzenia systemu ekspertowego. Wynika to głównie z tego, że inżynier wiedzy dokonuje wyboru metody reprezentacji na podstawie własnej intuicji i doświadczenia. Innym, dość istotnym problemem jest fakt, iż wiedza eksperta jest wiedzą o niektórych modelach procesów rzeczywistych, a nie o samej rzeczywistości. Poza tym część wiedzy eksperta może okazać się trudna do sformalizowania.

Implementacja wiedzy

Kolejnym etapem budowy systemu ekspertowego jest implementacja wiedzy, tzn. właściwe wykonanie systemu ekspertowego działającego na komputerze. Etap ten wymaga dogłębnej znajomości systemu szkieletowego, jakiego używa inżynier lub, jeśli tworzy on własny system, znajomości języka programowania, w którym system zostanie napisany. Oprócz poprawności merytorycznej, system powinien charakteryzować się łatwością obsługi, przejrzystością wprowadzanych danych oraz otrzymywanych wyników. Dobry interfejs z użytkownikiem ma również duże znaczenie komercyjne. Rezultatem implementacji jest prototypowy system ekspertowy, który podlega testowaniu i modyfikacjom. W przypadku doświadczonych inżynierów wiedzy proces implementacji bywa często łączony z procesem jej formalizacji.

Weryfikacja wiedzy

Metody testowania prototypowego systemu ekspertowego nie odbiegają w zasadzie od metod sprawdzania klasycznych programów komputerowych. Podczas fazy testowania ujawniają się niedociągnięcia i słabe punkty systemu. W wyniku ich wykrycia następuje powrót do jednego z poprzednich etapów budowy systemu celem dokonania poprawek. Testowania systemu powinien dokonywać ekspert, najlepiej inny niż ten, który brał udział w tworzeniu systemu. Pozwala to maksymalnie zobiektywizować proces testowania.

Podczas testowania systemu należy zwrócić uwagę na następujące czynniki [2, 10]:
- naturalność i jednoznaczność zadawanych przez system pytań,
- zgodność rozwiązań systemu z rozwiązaniami eksperta,
- czas wyprowadzania odpowiedzi,
- przejrzystość udzielanych odpowiedzi,
- przejrzystość udzielanych wyjaśnień.

Wykrycie błędu, zależnie od jego natury, prowadzi do powrotu do jednej z poprzednich faz projektowania, w najgorszym przypadku niektóre błędy działania systemu mogą wymagać powtórnej definicji problemu, co pociąga za sobą konieczność zmian we wszystkich etapach tworzenia systemu ekspertowego. Proces testowania jest długotrwały i pracochłonny.

Schematyczny proces budowy systemu ekspertowego przedstawia rys. 1.

Narzędzia tworzenia systemów ekspertowych

Spośród kilku dostępnych szkieletowych systemów ekspertowych, takich jak: Leonardo, IBIES-CX, GURU oraz języków wysokiego poziomu (LISP, PROLOG), do implementacji wiedzy o stanach wspomnianej na wstępie rozdzielni wybrano pakiet GURU. GURU [11] jest całym środowiskiem, służącym do tworzenia systemów ekspertowych. Będąc pakietem zintegrowanym GURU ma duże możliwości obróbki danych, takie jak:
- budowa i uruchamianie systemów ekspertowych,
- obsługa baz danych,
- stosowanie arkuszy kalkulacyjnych,
- obliczanie parametrów statystycznych,
- stosowanie graficznej reprezentacji danych,
- edycja tekstów,
- programowanie.

Wszystkie te możliwości mogą być używane jednocześnie w dowolnej kombinacji. Stwarza to możliwości tworzenia aplikacji od najprostszych baz danych, aż do skomplikowanych systemów wspomagania podejmowania decyzji. Użytkownik aplikacji stworzonych w środowisku GURU może pracować w trybie interakcyjnym nie tylko za pomocą interfejsu konkretnej aplikacji, ale również za pomocą standardowego interfejsu GURU, umożliwiającego konserwację w języku zbliżonym do naturalnego. Interfejs ten jest także wyposażony w procedury poszerzania słownika. Dzięki takiemu podejściu można zbudować złożone aplikacje, nadzorowane przez system ekspertowy albo też odwrotnie, aplikacja może się odwoływać do wielu różnych systemów ekspertowych celem rozwiązania zadania.

Pakiet GURU jest dostępny w wersjach dla różnych maszyn: od kompatybilnych z IBM PC do VaxMate. Pracuje on pod takimi systemami operacyjnymi, jak: DOS, OS/2 i UNIX, w wersjach zarówno jednostanowiskowych, jak i sieciowych. Istnieją dodatkowe biblioteki procedur umożliwiających dostęp do możliwości GURU z programów napisanych w C.

Kilka przykładowych cech GURU doskonale ilustruje duży stopień jego integracji:
- poprzednik dowolnej reguły może się odwoływać bezpośrednio do pól baz danych, komórek arkuszy kalkulacyjnych (również importowanych w standardzie Lotus 1-2-3), zmiennych statystycznych, zmiennych programowych oraz tablic;
- następnik reguły może zawierać instrukcje obsługi baz danych, arkuszy kalkulacyjnych, statystyki oraz generowania grafiki;
- zestaw reguł może wywoływać inne zestawy reguł, uruchamiać procedury i programy, korzystać z baz danych, przeprowadzać analizy statystyczne oraz prowadzić konwersację z użytkownikiem;
- programy mogą korzystać z komórek arkuszy kalkulacyjnych jako zmiennych, a nawet budować nowe arkusze,
- zawartość komórki arkusza kalkulacyjnego może pochodzić z bazy danych, być wynikiem działania programu lub wynikiem konsultacji systemu ekspertowego,
- zawartości komórek arkusza, wyniki działania programów, wyniki konsultacji systemów ekspertowych mogą być natychmiast umieszczone w tekście, który może podlegać dalszej obróbce lub drukowaniu.

Najważniejszym etapem w budowie systemu ekspertowego jest stworzenie bazy wiedzy. W GURU dokonuje się tego tworząc zestawy reguł. Definiowanie reguł może odbywać się za pomocą wbudowanego systemu tworzenia reguł albo też za pomocą wbudowanego (lub każdego innego eksportującego znaki ASCII) edytora tekstów.

Reguły są typowymi regułami produkcyjnymi IF-THEN. Poprzednik reguły praktycznie może być dowolnie skomplikowany. Może on zawierać pola baz danych, komórki arkuszy kalkulacyjnych, zmienne statystyczne, elementy tablic, konwencjonalne zmienne, skomplikowane wyrażenia matematyczne, wbudowane funkcje GURU, operatory logiczne (*AND, OR, XOR, NOT*), zmienne i wyrażenia rozmyte [3]. Następnik reguły może być sekwencją dowolnych poleceń systemu GURU. Każdej regule można ponadto przypisać priorytet, koszt wykonania, komentarz, objaśnienie, ciąg przygotowań poprzedzających sprawdzenie reguły. Zestawy reguł zawierają również opisy zmiennych oraz zestawy instrukcji wstępnych i końcowych.

Mechanizm wnioskowania stanowi integralną część pakietu GURU. Może on dokonywać zarówno rozumowania w przód, w tył, jak i w sposób mieszany. Dzięki zastosowaniu wielu opcji sterujących procesem wnioskowania, GURU może przeprowadzać proces poszukiwania rozwiązania z wykorzystaniem ponad 50 strategii przeszukiwań bazy wiedzy. Pozwala to zwiększyć sprawność i szybkość procesu wnioskowania.

Proces realizacji prototypu

Poszczególne etapy procesu realizacji prototypu systemu ekspertowego dla wspomnianej rozdzielni przebiegały następująco.

Zdefiniowanie problemu

Tworzony system ekspertowy na podstawie sygnałów widocznych w nastawni rozdzielni ma określić stan elementów rozdzielni oraz podać najbardziej prawdopodobne przyczyny wystąpienia w niej określonych stanów. Ze względu na obszerność wiedzy, zaimplementowany system został ograniczony do pola odpływowego. Wybór tego właśnie pola jest podyktowany faktem, że pól takich jest aż 22, na łączną liczbę 32 pól wszystkich typów w rozdzielni, a także względną złożonością obwodów sterowania, zabezpieczeń, sygnalizacji i blokad tego pola.

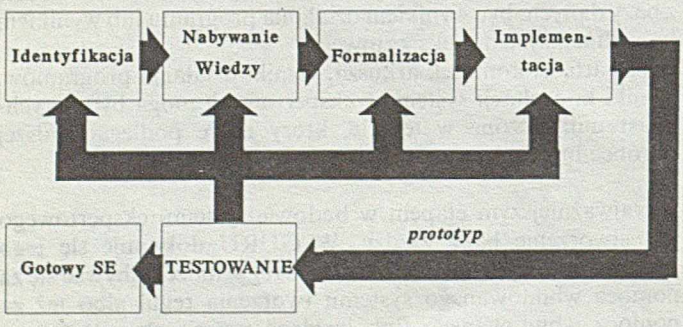
Gromadzenie wiedzy

Wiedza została zgromadzona na podstawie analizy schematów układów sygnalizacji, sterowania, zabezpieczeń i blokad pola odpływowego rozdzielni oraz na podstawie rozmów z kierownikiem rozdzielni RG-2, jako ekspertem z dziedziny stanów tej rozdzielni.

Celem stworzenia możliwie przejrzystej struktury systemu zastosowano następującą filozofię procesu konsultacji:

- określenie pola rozdzielni, którego dotyczy sygnał *GDZIE?*,
- określenie urządzenia, którego dotyczy sygnał *CO?*,
- określenie stanu tego urządzenia *SYTUACJA*,
- określenie szczegółów dotyczących istniejącej sytuacji, a umożliwiających określenie przyczyn wystąpienia konkretnego stanu rozdzielni *DETALE*.

Schemat podejmowania decyzji wraz z relacjami między zestawami reguł pokazuje rysunek 2.



Rys. 2. Zależności pomiędzy zestawami reguł prototypu SE

Formalizacja i implementacja wiedzy

Zgromadzona wiedza została sformalizowana oraz zaimplementowana w środowisku narzędziowym GURU. Baza wiedzy została utworzona na podstawie wcześniej opisanego schematu konsultacji.

Weryfikacja wiedzy

Proces weryfikacji wiedzy polegał na przedstawieniu kierownikowi rozdzielni wiedzy zgromadzonej na podstawie analizy schematów rozdzielni. Wiedza ta nie wymagała większych modyfikacji. Jedynie w kilku przypadkach okazało się, że na stany w polu odpływowym mają wpływ stany innych pól rozdzielni. Inne korekty wynikały ze specyfiki działania urządzeń, której nie można było odkryć ze schematów elektrycznych. Dalsze nieścisłości i niezgodności z rzeczywistością mogą być więc wykryte dopiero podczas intensywnego testowania prototypu.

* * *

Podczas wstępnego testowania systemu prototyp wykazywał poprawną pracę i na postawione pytania dawał odpowiedzi podobne do udzielonych przez eksperta. Niewielkie różnice wynikały z faktu, że baza wiedzy prototypu zawiera wiedzę dotyczącą tylko pola odpływowego. Natomiast ekspert, dokonując analizy, korzysta również z wiedzy o innych elementach rozdzielni.

W procesie weryfikacji wiedzy system wykazał dużą elastyczność na dokonywanie zmian. Usuwanie lub dodawanie nowych reguł lub relacji nie stwarzało kłopotów ani nie powodowało wadliwej pracy systemu. Pozwalało to łatwo uzupełniać bazę wiedzy prototypu i rozszerzyć analizę na pozostałe pola rozdzielni.

Dobre odzwierciedlenie rzeczywistości pozwala zastosować obecną wersję systemu w procesie szkolenia operatorów rozdzielni w zakresie określenia przyczyn podstawowych stanów rozdzielni oraz ewentualnych uszkodzeń. Uzupełnienie reguł o komentarze i szersze objaśnienia pozwolą poprawić i przyspieszyć proces szkolenia.

Właściwości środowiska GURU pozwalają na jego komunikowanie się z systemami zewnętrznymi oraz programami napisanymi w języku C. Umożliwia to dostosowanie prototypu do pracy w trybie on-line. Wymaga jednak stworzenia odpowiedniego urządzenia sprzęgającego komputer z obwodami sygnalizacji, sterowania i blokad rozdzielni.

Przystosowanie prototypu systemu do pracy w warunkach przemysłowych wymaga jego rozbudowy, a głównie uzupełnienia bazy wiedzy. Konieczne jest też przeprowadzenie długotrwałego procesu jego testowania i weryfikacji.

LITERATURA

[1] GURU Expert Systems Management. Ver. 2.0, mdbs Inc., 1989
[2] Klir G., Folger T.: Fuzzy Sets, Uncertainity and Information. Prentice-Hall, 1988

dokończenie na s. 24

BEATA CZARNACKA-CHROBOT
Katedra Systemów Informatycznych
Szkoła Główna Handlowa
Warszawa

Metoda FUNCTION POINTS

Jednostki zajmujące się projektowaniem systemów informatycznych dysponują zarówno zasobami komputerowymi, jak i ludzkimi. Tworząc system aplikacyjny projektanci powinni zagwarantować, że środki te będą wykorzystane właściwie: efektywnie, wydajnie i przyniosą zysk. W szczególności każdy zespół projektujący powinien wykazać, że:

- Zajmuje się udoskonalaniem sposobu wykonywania funkcji użytkownika przyszłego systemu. Dla użytkownika wartość aplikacji niej polega na liczbie linii kodu programu, ale na sposobie, w jaki system informatyczny zaspokoi zadania bezpośrednich użytkowników oraz w jakim stopniu przyczyni się on do zwiększenia zysków firmy użytkownika. Podstawowymi żądaniami wysuwanymi wobec projektantów zajmujących się konstrukcją aplikacji jest dostarczenie systemu informatycznego na czas i mieszczącego się w budżecie, jaki został przeznaczony na stworzenie takiego systemu. Spełnienie tych postulatów wymaga istnienia efektywnej metody kontroli projektu, dzięki której już we wstępnych fazach projektowania można oszacować koszt, a zatem i cenę, projektu.
- Inwestowanie w działalność jednostek tworzących i rozwijających systemy informatyczne jest uzasadnione, gdyż ich praca przynosi zysk firmie, w której skład te jednostki wchodzą.

Powyższe czynniki determinują fakt, że zarówno wartość tworzonego systemu, jak i niezbędne do jego utworzenia nakłady projektantów powinny być możliwe do oszacowania już na początku prac nad systemem informatycznym. Sprowadza się to do pomiaru **produktywności pracy** projektantów przez zespół zajmujący się projektowaniem.

Zastosowanie procedur mierzenia produktywności tworzenia i rozwoju aplikacji było w ostatnich latach tematem wielu rozważań. Jednakże wszystkie dotychczasowe sposoby pomiaru tej wielkości okazywały się mało efektywne. W IBM stworzono metodę opartą, ogólnie rzecz ujmując, na analizie tych funkcji użytkownika, których realizację ma zapewnić wprowadzany system komputerowy. Ta metoda pomiaru jest znana pod nazwą analizy *FUNCTION POINTS*[1].

Podstawowymi celami metody *Punktów Funkcyjnych* są:

- ocena zrealizowanych projektów oraz wyprowadzenie na tej podstawie wielkości nakładów pracy projektantów, jakie są niezbędne do realizacji tych projektów;
- rozpoznanie oraz przedstawienie trendów produktywności pracy projektantów;
- obserwowanie efektywności bieżącej pracy projektantów.

Produktywność projektowania systemów informatycznych

Produktywność jest pojęciem związanym z produktem (końcowym efektem prac) oraz z możliwościami czynnika ludzkiego, tj. nakładami pracy, jakie są niezbędne do stworzenia zaplanowanego systemu informatycznego.

Praca projektantów systemów informatycznych dotyczy dwóch zasadniczych obszarów:

- budowy i dostarczenia nowego systemu zaspokajającego zdefiniowane funkcje użytkownika.
- utrzymania funkcjonującego już systemu, przy czym w przypadku pomiaru produktywności, istotne znaczenie ma następujące rozróżnienie prac związanych z utrzymaniem istniejącego systemu:

– prac polegających na jego doskonaleniu, a więc zmierzających do rozwoju aplikacji,
– prac polegających jedynie na ciągłym podtrzymywaniu jego pierwotnej funkcjonalności – działanie takie nazwiemy konserwacją systemu.

W związku z powyższym rozróżnia się dwa rodzaje produktywności projektowania systemu informatycznego:

- Produktywność rozwoju aplikacji;

– stosunek wartości produktu nowego lub udoskonalonego do nakładów pracy włożonych w budowę nowego lub udoskonalenie już istniejącego systemu,
– odnosi się zarówno do nowych, jak i udoskonalania już istniejących aplikacji (pod warunkiem, że system zmienia funkcje użytkownika).

- Produktywność konserwacji aplikacji:

– stosunek wartości całego systemu, który jest utrzymywany, do nakładów pracy włożonych w konserwację systemu,
– odnosi się do prac konserwacyjnych, na które składa się każda praca, niezbędna do tego, aby system w dalszym ciągu zapewniał użytkownikom te same usługi (dotyczy jedynie systemów już funkcjonujących).

Oczekiwania w stosunku do systemu pomiaru produktywności

System pomiaru produktywności powinien realizować następujące cele:

- zapewniać możliwość porównywania różnych projektów,
- zapewniać możliwość porównywania efektów pracy jednostek[2] zajmujących się projektowaniem aplikacji,
- umożliwiać wyznaczanie tendencji produktywności pracy zespołów projektujących,
- wspomagać przez pomiar produktywności i jej analizę lub decyzje, które mogą udoskonalić rezultat pracy poszczególnych jednostek zajmujących się tworzeniem projektu,
- powinien służyć użytkownikom w zrozumieniu natury ich potrzeb, które w formie zadań przekazują projektantom,

[1] Twórcą tej metody (1981) jest Allan J. Albrecht, pracownik IBM w USA. W ostatnich latach została ona rozwinięta przez P. Ruede. W niniejszym artykule używany będzie polski odpowiednik: *metoda Punktów Funkcyjnych.*

[2] Jednostka (ang. *site*) jest tu rozumiana jako oddzielnie zarządzany dział przedsiębiorstwa. W ramach jednego przedsiębiorstwa może istnieć wiele różnych jednostek zajmujących się tworzeniem projektów systemów informatycznych.

- umożliwiać pomiar w takiej formie, aby użytkownik, nie będący ekspertem w dziedzinie techniki komputerowej mógł go zrozumieć i powiązać z innymi wielkościami,
- zapewniać przydatność stosowanej metody we wszystkich fazach, zarówno projektowania nowej aplikacji, jak i utrzymania istniejącego systemu; metoda ta powinna dotyczyć całego cyklu życia systemu, począwszy od zdefiniowania zadań a skończywszy na jego utrzymaniu tak, aby – jeżeli to konieczne – można było oszacować produktywność poszczególnych faz,
- powinien zapewniać możliwość wyznaczania miary produktywności we wczesnych fazach cyklu rozwoju aplikacji.

Wymagania systemu pomiaru produktywności

Sprostanie wyżej wymienionym celom stwarza konieczność, aby wszyscy członkowie zespołów projektujących byli zainteresowani zastosowaniem systemu pomiaru produktywności. Ponadto, system pomiaru produktywności wymaga:
- spójności definicji – produktywność jest definiowana jako relacja produktu pracy do nakładów pracy. Produkt pracy oraz nakłady pracy powinny być właściwie i spójnie zdefiniowane tak, aby poszczególne czynności można było w analizie jednoznacznie zidentyfikować;
- wysokiej jakości zarządzania – wszystkie zadania stojące przed projektantami są od siebie współzależne. Jakość i kompletność wykonania zadania wcześniejszego może wpływać znacząco na produktywność późniejszych zadań. Proces zarządzania oraz kontrolowania wszystkich zadań, faz i podprojektów powinien zapewniać, że wynik każdej czynności zaspokaja jakościowo wyspecyfikowane dla tej czynności żądania;
- technologicznej elastyczności – aby uniknąć zależności od używanych w procesie projektowania metod, system pomiaru produktywności powinien mierzyć produkt pracy w kontekście funkcji dostarczanych użytkownikom[3];
- uczestniczenia użytkownika systemu w kalkulacji produktu pracy;
- dyscypliny obliczeń.

Pomiar produktywności a estymacja

Podstawową zaletą metody *Punktów Funkcyjnych* jest możliwość oszacowania rozmiaru produktu pracy, zarówno w kontekście procesu estymacji, jak i pomiaru produktywności. Rysunek pokazuje powiązania między estymacją a pomiarem produktywności oraz dane, jakie są potrzebne dla każdego z tych działań.

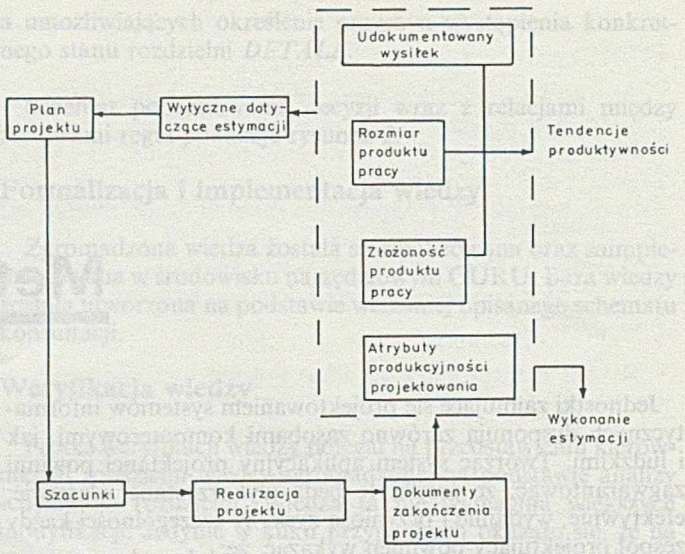
Pomiar produktu pracy powstającego w procesie projektowania aplikacji

Punktem wyjścia dla stworzenia metody Punktów Funkcyjnych była analiza procesu przetwarzania informacji.

Elementy metody Punktów Funkcyjnych

Wśród elementów związanych z przetwarzaniem informacji wyróżniono te, które mają charakter kwantytywny. Elementami tymi są:
- wejście – są to dane, które użytkownik zamierza wprowadzać do systemu,
- wyjście – są to informacje, które będą generowane przez system,

[3] Definiowanie produktu pracy powinno odbywać się raczej w kontekście jego funkcjonalności, a nie w liniach kodu źródłowego.



Związek między estymacją a pomiarem produktywności [1]

- logiczne zbiory wewnętrzne – są to dane, które pozwalają systemowi na generowanie wyróżnionych informacji wyjściowych, zawierających informacje użytkownika,
- zewnętrzne zbiory komunikacyjne (ang. *interface files*) – są to dane umożliwiające systemowi przekazywanie informacji do innego systemu,
- zapytania – są to zapytania, które użytkownik będzie wprowadzał do systemu w celu uzyskania na nie odpowiedzi.

Wszystkie wymienione elementy są głównymi typami danych systemu. Mają one istotne znaczenie dla określenia względnej wartości procesu przetwarzania danych na trzech poziomach złożoności: prostym, średnim oraz kompleksowym. Poddano je badaniu celem określenia wag poszczególnych typów danych w zależności od poziomu złożoności procesu ich przetwarzania (patrz tab. 1).

Tabela 1. Wagi typów danych według poziomów złożoności [1]

Poziom złożoności / Typ danych	Prosty	Średni	Złożony
Wejścia użytkownika	3	4	6
Wyjścia użytkownika	4	5	7
Wewnętrzne zbiory logiczne	7	10	15
Zewnętrzne zbiory komunikacyjne	5	7	10
Zewnętrzne zapytania użytkownika	3	4	6

W projektowanym systemie należy wyróżnić wszystkie typy danych, określić dla każdego z nich poziom złożoności przetwarzania, a następnie przypisać im, wynikające z tabeli 1, wagi. Kolejnym krokiem jest zsumowanie wag.

Suma ważonych typów danych charakterystycznych dla danego systemu daje całkowity, ale nie ostateczny (ang. *unadjusted*) wynik metody Punktów Funkcyjnych. Rezultat ten ma jedynie charakter pośredni, gdyż wymaga poprawki ze względu na inne czynniki występujące w projektowanym systemie, tzw. parametry wpływu. Stosuje się je jako korektę poprzednich wyników, gdyż mają one również wpływu na prace projektowe. Do parametrów wpływu zalicza się:
- występowanie urządzeń komunikacyjnych,
- występowanie funkcji rozproszonych,
- długość czasu oczekiwania na odpowiedź systemu,

- stopień obciążenia systemu,
- częstotliwość wykonywania transakcji
- występowanie wprowadzania danych w trybie on-line,
- wydajność użytkownika końcowego,
- występowanie aktualizacji danych w trybie on-line,
- stopień złożoności przetwarzania,
- możliwość ponownego użycia programów w innych zastosowaniach,
- łatwość instalowania, tj. konwertowanie (łatwość zmiany formatu),
- łatwość obsługi,
- wielokrotność lokalizacji, tj. występowanie stacji lokalnych,
- łatwość wprowadzania zmian

Każdy z czternastu wyżej wymienionych parametrów może w danym systemie występować i mieć w nim zróżnicowany wpływ. Dlatego zdecydowano się określić ilościowo stopień, do jakiego charakterystyki te mogą wpływać na system:

- nie reprezentowane lub bez wpływu = 0
- wpływ nieznaczny = 1
- wpływ umiarkowanny = 2
- wpływ średni = 3
- wpływ znaczny = 4
- wpływ silny = 5

Każdemu z charakterystycznych dla danego systemu parametrów wpływu przypisuje się stopień wpływu. W wyniku zsumowania stopni wpływu otrzymujemy tzw. całkowitą wartość wpływu, będącą wskaźnikiem korygującym. Stosując odpowiednie wzory wskaźnik ten wykorzystuje się w odniesieniu do pośredniego rezultatu pomiaru metodą Punktów Funkcyjnych celem uzyskania rezultatu końcowego. Rezultat ten spełnia wyżej przedstawione wymagania, jakie wysuwa się w stosunku do systemu pomiaru produktywności.

Kalkulacja punktów funkcyjnych

Jak już zaznaczono, można wyróżnić dwa rodzaje produktywności projektowania systemów informatycznych, a zatem i dwa rodzaje produktu pracy projektantów:

- system nowo powstający lub udoskonalony,
- system utrzymywany w sensie jego konserwacji.

W związku z tym ustalono dwa równania, służące wyliczeniu punktów funkcyjnych w odniesieniu do:

- systemu nowo powstającego lub udoskonalanego (PPN/D – nowy lub udoskonalony produkt pracy),
- produktu pracy projektantów mającego na celu konserwację (w sensie utrzymania dotychczasowej liczby punktów funkcyjnych) systemu już wykorzystywanego (PPK – produkt pracy konserwacyjnej).

Równanie 1

Ostateczna liczba punktów funkcyjnych dla PPN/D:

$$PF^{4} = (A + B1) \times (C2) + (D) \times (C1)$$

Równanie 2

Ostateczna liczba punktów funkcyjnych dla PPK:

$$PF = E + (A + B1) \times (C2) - (D + B2) \times (C1)$$

[4] Dla nowego projektu będzie to całkowita, nieostateczna liczba punktów funkcyjnych, pomnożona przez czynnik korygujący ze względu na złożoność przetwarzania, czyli: $PF = A \times C2$

Objaśnienie użytych we wzorach symboli literowych:

PF – liczba punktów funkcyjnych po skorygowaniu (ostateczna),

A – niedostateczna liczba punktów funkcyjnych **dodanych** do aplikacji,

$B1$ – nieostateczna liczba punktów funkcyjnych **zmienionych** w aplikacji, obliczana przy zakończeniu projektu (nie dotyczy aplikacji nowo tworzonych),

$B2$ – nieostateczna liczba punktów funkcyjnych **zmienionych** w aplikacji, szacowana przed rozpoczęciem działań konserwacyjnych,

$C1$ – czynnik korygujący (dostosowanie ze względu na złożoność przetwarzania), szacowany **przed** rozpoczęciem projektowania; dotyczy aplikacji już wdrożonych,

$C2$ – czynnik korygujący, tj. dostosowanie ze względu na złożoność przetwarzania, właściwe dla aplikacji **po** zakończeniu prac projektowych; dotyczy również aplikacji nowo powstałych,

D – nieostateczna liczba punktów funkcyjnych **usuniętych** z aplikacji, szacowana przed rozpoczęciem projektowania; nie dotyczy aplikacji nowo powstających,

E – ostateczna liczba punktów funkcyjnych aplikacji przed rozpoczęciem działań konserwacyjnych[5].

Możliwe jest wykorzystanie mniej niż czternastu atrybutów – stworzy to mniej prawidłową reprezentację produktywności danego systemu. W takim przypadku powinno ulec zmianie obliczanie czynnika korygującego ze względu na złożoność przetwarzania C, wykorzystywanego do wyprowadzenia liczby punktów funkcyjnych[6].

W obu przypadkach w praktyce stosuje się zazwyczaj mniej skomplikowaną formułę[7], ujętą w postaci równania 3.

Równanie 3

Ma ono następującą postać[8]:

$$PF = NPF \times C$$

[5] Jeżeli aplikacja składa się jedynie z nowych lub dodanych typów funkcji, to liczba punktów funkcyjnych dla PPN/D (wtedy PPN) równa się liczbie punktów funkcyjnych dla PPK. Wówczas dla równania 1: $PF = A \times C2$, natomiast dla równania 2: $C2 = 0$, z czego wynika, że $PF = A \times C2 = E$. Jeżeli mamy do czynienia z aplikacją udoskonaloną, obejmującą dodane, usunięte i/lub zmienione typy funkcji, liczba punktów funkcyjnych dla PPN/D (wtedy PPD) nie równa się liczbie punktów funkcyjnych dla PPK.

[6] Zazwyczaj stosuje się następującą formułę:
$C = 0{,}65 + \langle 0{,}01 \times F \rangle$, gdzie F oznacza złożoność przetwarzania, otrzymywaną jako suma wszystkich stopni wpływów głównych parametrów aplikacji. Liczbą dostosowującą jest 0,65. Była ona wyprowadzona w sposób następujący:

- założono, że czynnik korygujący C dla średniego projektu powinien wynosić 1,
- następnie obliczono średnią złożoność przetwarzania F: $14 \times 2{,}5 = 35$,

gdzie mamy 14 atrybutów ze średnim stopniem wpływu 2,5.

- Z równania:

$1 = x + (0{,}01 \times 35)$ otrzymujemy $x = 0{,}65$.

Np. zmiana atrybutów na 10, oznacza zmianę tej formuły na:
$C = 0{,}75 + (0{,}01 \times F)$, ponieważ $F = 10 \times 2{,}5 = 25$
Możliwe jest też zmienienie stopni wpływów. Można np. przyjąć, że dla ułatwienia pomiarów przyjmujemy zamiast wartości 0–5 wartości binarne (choć nie jest to rekomendowane):
0 oznacza brak wpływu,
1 oznacza istniejący wpływ.
W takiej sytuacji średni wpływ wynosi 0,5, a formuła przy użyciu czternastu atrybutów będzie miała postać:
$C = 0{,}93 + (0{,}01 \times F)$, ponieważ $F = 14 \times 0{,}05 = 7$.

[7] Przyjmuje się tu założenie, że $C1$ jest w przybliżeniu równe $C2$ i określa się je jako C.

przy czym $C = (0{,}65 + F \times 0{,}01)$, czyli
$PF = NPF \times (0{,}65 + F \times 0{,}01)$,

gdzie:
NPF – nieostateczna liczba punktów funkcyjnych obliczana różnie, w zależności od rodzaju działalności projektowej[9],
F – złożoność przetwarzania, będąca sumą wszystkich stopni wpływów głównych atrybutów aplikacji,
0,65 – liczba dostosowywująca.

Pomiar nakładów pracy na utworzenie, rozwój oraz konserwację systemu

Celem niniejszego artykułu jest przybliżenie czytelnikowi metody Function Points, służącej do pomiaru jednej z dwóch wielkości składających się na produktywność projektowania, mianowicie produktu pracy. Konieczność pomiaru drugiej wielkości, tzn. nakładów pracy, zostanie tylko zasygnalizowana.

Jeżeli wysiłek włożony w pracę nad systemem nie zostanie kompletnie i właściwie obliczony i dokumentowany, to wyniki dotyczące produktywności będą błędne i wszystkie oparte na takich danych szacunki będą niewiarygodne.

Cele systemu pomiaru nakładów pracy

System pomiaru nakładów pracy, aby mógł wspomagać zarówno pomiar produktywności, jak i procesy estymacyjne, powinien spełniać następujące cele:
- pokrywać cały cykl życia systemu,
- być zgodny dla wszystkich projektów,
- definiować standardowe zadania dla każdej fazy,
- przewidywać niestandardowe czynności wszędzie tam, gdzie są one niezbędne,
- dostarczać dostatecznych szczegółów do wspomagania procesów estymacyjnych,

Miary nakładów pracy projektantów

Należy brać pod uwagę globalne nakłady pracy użyte do zrealizowania zadań oraz wykonania czynności i faz odnoszących się do danego procesu projektowania[10]. Nakłady pracy powinny uwzględniać osobogodziny, osobomiesiące oraz osobolata, wykorzystane do realizacji następujących faz procesów związanych z projektowaniem:
- **właściwego projektowania,** w skład którego wchodzą następujące etapy:
  - studia (tzw. badanie wstępne),
  - forumułowanie wymagań użytkownika, czyli projektowanie zewnętrzne (tzw. ogólnokoncepcyjne),
  - projektowanie wewnętrzne (tzw. techniczne),
  - zaplanowanie implementacji (fazy następnej);
- **implementacji,** w skład której wchodzą następujące etapy:
  - programowanie (projektowanie programów), kodowanie oraz testowanie jednostek (tzw. modułów),
  - integracja modułów,
  - testowanie całości oprogramowania,
  - dokumentowanie,
  - planowanie instalacji i utrzymywania;
- **instalacji,** obejmujący etapy:
  - szkolenia użytkownika,
  - konwersję danych (właściwą instalację systemu);
- **utrzymania,** w skład którego wchodzą następujące etapy:
  - wspomaganie zainstalowanej aplikacji,
  - stałe utrzymywanie jej funkcjonowania,
  - retrospektywne analizy i ewentualne modyfikacje.

Istnieją dwie miary nakładów pracy:
- czas pracy netto,
- czas pracy brutto.

Czas pracy netto odnosi się do czasu aktualnie wydatkowanego na pracę nad zadaniami i czynnościami dotyczącymi projektu. Wyłącza się z niego, w przeciwieństwie do czasu pracy brutto, czas przeznaczony na urlopy, choroby, kształcenie oraz inne nieobecności.

Podstawową jednostką pomiaru jest godzina pracy brutto (gpb), która jest definiowana jako *godzina możliwa (dostępna) do przepracowania dla jednej osoby* (osobogodzina brutto). Opierając się na tej definicji można zdefiniować agregatowe miary czasu (osobodni, -miesiące, -lata). Jednakże najlepszą jednostką pomiaru jest godzina pracy netto (osobogodzina netto). Wszystkie inne miary używane do pomiaru nakładów pracy powinny być odnoszone do tej jednostki. Odzwierciedla ona czas faktycznie wydatkowany na czynności związane z działalnością projektową.

* * *

Pomiar produktywności w odniesieniu do systemów nowo tworzonych oraz tych, które zostały już wdrożone, ułatwia identyfikację i zatwierdzanie korzystnych zmian oraz ilustruje wartość produktu pracy jednostek zajmujących się projektowaniem. Ponadto pomiar dostarcza podstawowych danych do wspomagania procesów estymacyjnych i kontrolnych dotyczących projektu tak, aby w końcowym efekcie projektanci wypełnili swoją rolę i dostarczyli użytkownikowi aplikacje zgodnie z zaplanowanym czasem i w ramach zaplanowanych środków. Chociaż również do mierzenia produktu pracy projektantów mogą być używane linie kodu, to jednak miara ta ma istotne ograniczenia. Metodę *Function Points* zaleca się dlatego, gdyż ma ona bardziej uniwersalne zastosowanie oraz może być wykorzystywana przed rozpoczęciem szczegółowego projektowania (w fazie projektowania zewnętrznego).

Reasumując, można wyznaczyć następujące ogólne stadia (kroki) pomiaru produktywności opartej na metodzie *Punktów Funkcyjnych*[11].
1. Wyróżnienie modułów projektu.
2. Określenie, w stosunku do każdego modułu, tzw. głównych typów danych (wejść, wyjść, logicznych zbiorów wewnętrznych, zbiorów komunikacyjnych oraz zapytań).
3. Ocena wyżej wymienionych typów danych oraz przypisanie im na podstawie standardowej tabeli wag w zależności od poziomu złożoności, co prowadzi do obliczenia niedostatecznej liczby punktów funkcyjnych.

---

[8] Maksymalna złożoność przetwarzania F dla projektu wynosi 70 (14×5), natomiast minimalna – 0, i dlatego średnia złożoność wynosi 35. Przedstawiona formuła zapewnia, że czynnik dostosowujący ze względu na złożoność przetwarzania C dla „średniego" projektu wynosi 1.

[9] Dla *PPN:*
$B1 = 0$ oraz $D = 0$, a zatem $PF = C \times A$, czyli $NPF = A$
Dla *PPD:*
$PF = (A + B1) \times (C2) + (D) \times (C1)$, ponieważ $C1 = C2$, to
$PF = (A + B1 + D) \times C$, a zatem $NPF = A + B1 + D$.
Dla *PPK:*
$PF = E + (A + B1) \times C2 - (D + B2) \times (C1)$, ponieważ $C1 = C2$, to
$PF = E + (A + B1 - D - B2) \times C$, a zatem $NPF = A + B1 - D - B2$.

[10] Jeżeli np. użytkownik pracuje nad zadaniami dotyczącymi projektowania, to również należy uwzględnić czas użytkownika przy obliczaniu całkowitych nakładów pracy dla danego projektu.

[11] Przy czym sama metoda Punktów Funkcyjnych dotyczy kroków 1–5.

4. Analiza parametrów wpływu oraz określenie stopnia ich wpływu, co stanowi podstawę do wyliczenia ostatecznej liczby punktów funkcyjnych (tj. korygowania nieostatecznej liczby punktów funkcyjnych ze względu na złożoność przetwarzania).
5. Obliczenie ostatecznej liczby punktów funkcyjnych wg wyżej przedstawionych wzorów.
6. Określenie nakładów pracy projektantów w osobojednostkach.
7. Obliczenie produktywności, tj. stosunku ostatecznej liczby punktów funkcyjnych (wyrażających „wartość" systemu) do nakładów pracy (wyrażających „koszt" projektowania).

Przykład zastosowania metody Punktów Funkcyjnych[12)]

Krok 1

Wyróżnienie modułów obsługujących poszczególne funkcje pewnego przedsiębiorstwa handlowego. Są one następujące:
1. Zapytania i opracowanie proponowanego asortymentu.
2. Kontrola proponowanych ofert.
3. Przyjmowanie zleceń i ich kontrola.
4. Dyspozycja przesyłaniem.
5. Fakturowanie.
6. Zapisanie na dobro rachunku (księgowanie).

Krok 2

Dla poszczególnych modułów wyróżniono następujące typy danych:

Ad.1.

Wejście:	Zapytania klientów	(kompleksowe)	(6)
	Parametry akcji reklamowych (ofert)	(proste)	(3)
Wyjście:	Indywidualne propozycje sprzedaży (oferty)	(średnie)	(5)
	Selektywnie prowadzone masowe propozycje sprzedaży	(średnie)	(5)
Zbiór danych:	Klienci		
	Wyroby		
	Struktury wyrobów		
	Oferty		

Ad. 2.

Wejście:	Instrukcja kasowania	(proste)	(3)
Wyjście:	Przypominanie o ofertach	(proste)	(4)
	Protokół kasowania	(proste)	(4)
Zapytanie:	Aktualne oferty	(proste)	(3)
Zbiór danych:	Klienci		
	Wyroby		
	Oferty		

Ad. 3.

Wejście:	Zapytania klientów	(kompleksowe)	(6)
	Storno	(średnie)	(4)
Wyjście:	Odrzucone zlecenia	(średnie)	(5)
	Potwierdzenie zleceń	(kompleksowe)	(7)
	Stornowanie zleceń	(średnie)	(5)
	Fakturowanie dla stanu magazynowego	(kompleksowe)	(7)
Zapytania:	Aktualne zlecenia	(średnie)	(4)
Zbiór danych:	Klienci		
	Wyroby		
	Oferty		
	Zlecenia		

Ad. 4.

Wejście:	Nakaz szukania	(proste)	(3)
Wyjście:	Wskazanie przesyłania	(proste)	(4)
	Dokumenty frachtu	(proste)	(4)
	Dokumenty dostawy	(proste)	(4)
	Naklejki	(proste)	(4)
Zbiór danych:	Klienci		
	Dostawcy		
	Wyroby		
	Zlecenia		

Ad. 5.

Wejście:	Dokumenty przesyłania	(średnie)	(4)
Wyjście:	Rachunki	(proste)	(4)
	Obciążenia	(proste)	(4)
	Spis wyjść rachunków	(proste)	(4)
	Fakturowanie dla księgowości	(proste)	(4)
Zbiór danych:	Klienci		
	Wyroby		
	Materiały do pakowania		
	Zlecenia		
	Agenci		

Ad. 6.

Wejście:	Rachunki wartościowe	(średnie)	(4)
	Rachunki powrotne	(średnie)	(4)
Wyjście:	Rachunki	(proste)	(4)
	Zalecenia księgowania	(proste)	(4)
Zbiór danych:	Klienci		
	Wyroby		

Zbiory danych	Punkty funkcyjne
Klienci (dwie logiczne grupy danych), 2 × 7	(14)
Wyroby (dwie logiczne grupy danych), 2 × 7	(14)
Struktury wyrobów	(15)
Dostawcy	(7)
Materiały do pakowania	(7)
Agenci	(7)
Oferty	(10)
Zlecenia (dwie logiczne grupy danych), 2 × 7	(14)

Krok 3

W zależności od poziomu złożoności, przypisanie wag poszczególnym typom danych następuje na podstawie podanego w standardowej tabeli 1 wzorca (tab. 2).

Tabela 2. Przykładowe przypisanie wag poszczególnym typom danych

Opis	Złożoność			
	Prosta	Średnia	Kompleksowa	Suma
Wejście	3 × 3 = 9	4 × 4 = 16	2 × 6 = 12	37
Wyjście	12 × 4 = 48	4 × 5 = 20	2 × 7 = 14	82
Logiczne zbiory wewnętrzne	9 × 7 = 63	1 × 10 = 10	1 × 15 = 15	88
Zewnętrzne zbiory komunikacyjne	0 × 5 = 0	0 × 7 = 0	0 × 10 = 0	0
Zapytania	1 × 3 = 3	1 × 4 = 4	0 × 6 = 0	7
Całkowita niedostateczna liczba punktów funkcyjnych (NPF)				214

[12)] Przykład dotyczy systemu nowo powstającego.

Krok 4

Tabela 3. Analiza parametrów wpływu wraz z określeniem stopnia ich wpływu w skali 0–5

Lp.	Parametr	Stopień wpływu	Wartość
1.	Występowanie urządzeń komunikacyjnych	średni	3
2.	Występowanie funkcji rozproszonych	brak	0
3.	Czas oczekiwania na odpowiedź	średni	3
4.	Stopień obciążenia systemu	brak	0
5.	Częstotliwość wykonywania transakcji	nieznaczny	1
6.	Wprowadzenie danych w trybie on-line	silny	5
7.	Wydajność użytkownika końcowego	umarkowany	2
8.	Aktualizacja danych w trybie on-line	nieznaczny	1
9.	Stopień złożoności przetwarzania	umiarkowany	2
10.	Możliwość ponownego użycia programów w innych zastosowaniach	nieznaczny	1
11.	Łatwość instalowania	brak	0
12.	Łatwość obsługi	średni	3
13.	Wielokrotność lokalizacji	brak	0
14.	Łatwość wprowadzania zmian	umiarkowany	2
Całkowita wartość wpływu (F)			**23**

Krok 5

Obliczenie ostatecznej liczby punktów funkcyjnych na podstawie równania 3:

$PF = NPF \times C$ gdzie:

$C = 0{,}65 + (0{,}01 \times F) = 0{,}65 + (0{,}01 \times 23) = 0{,}88$,
a zatem:

$PF = 214 \times 0{,}88 = 188$

Krok 6

Tabela 4. Określenie nakładu pracy projektantów na podstawie wieloletnich badań projektów

Ostateczna liczba Punktów Funkcyjnych	Osobomiesiące netto
50	2,3
100	5,6
150	9,5
200	13,9
250	18,6
300	23,6
350	28,9
400	34,4
450	40,1
550	46,1
550	52,2
600	58,5
650	65,0
700	71,6
750	78,4
800	85,3
850	92,4
900	99,6
950	106,9
1000	114,4
1100	129,6
1200	145,2
1300	161,3
1400	177,7
1500	194,6
1600	211,7
1700	229,3
1800	247,1

Określenie nakładu pracy projektantów, do którego wykorzystuje się w praktyce tabelę 4, powstałą na podstawie wieloletnich badań.

Ostateczna liczba punktów funkcyjnych wyniosła 188, a zatem na podstawie tabeli 4 można oszacować, że nakłady pracy netto, niezbędne do wykonania systemu prezentowanego w przykładzie, wynoszą około 12,5 osobomiesięcy[13].

Krok 7

Obliczenie produktywności projektowania omawianego systemu:

$188 : 12{,}5 = 15{,}04$

Jeżeli po zakończeniu działań projektowych okaże się, że uzyskano liczbę większą oznacza to, że wzrosła produktywność pracy projektantów. Natomiast, jeżeli zostanie uzyskana liczba mniejsza – oznacza że na realizację projektu wydatkowano zbyt duże nakłady pracy.

Wielkość tę można również porównywać z innymi projektami wykonywanymi w ramach tej samej jednostki projektującej, badając tym samym zarówno jej kondycję, jak i rozbieżności między różnymi jednostkami zajmującymi się tworzeniem systemów informatycznych.

LITERATURA

[1] Development Centre Productivity Measurement Guide. IBM 1984

[13] Z obliczeń wynika, że wielkość nakładów pracy waha się między 12 a 13 osobomiesięcy; dlatego przyjęto wynik średni 12,5.

Budowa prototypu systemu ekspertowego

dokończenie ze s. 18

[3] Kosmowska-Miszalska D.: Strojenie mechanizmu rozumowania systemów ekspertowych z uwzględnieniem warunków niepewności. INFORMATYKA nr 9,1992
[4] Liang T.P.: Automated Knowledge Acquisition. Morgan Kaufman, 1992
[5] Owoc M.L.: Elementy systemów ekspertowych, Wrocław, 1991
[6] Pham D.T.: Expert Systems in Engineering, IFS Publications, Springer-Verlag, 1988
[7] Ramsay A.: Formal Methods in Artificial Intelligence. Cambridge University Press, 1988
[8] Shapiro S.U.: Encyklopedia of Artificial Intelligence. John Wiley & Sons, New York, 1987
[9] Stefik M.: The Organization of Expert Systems, A. Tutorial. Artificial Inteligence, March 1982
[10] Turban E.: Expert Systems and Applied Artificial Intelligence, Macmillan Publishing Company, 1992
[11] Waterman D: A Guide to Expert Systems, Addison-Wesley, 1986.

Ogłoszenia prosimy zgłaszać:

pisemnie
Red. INFORMATYKI
Pl. Inwalidów 10 p. 104, 01-522 Warszawa
telefonicznie
nr 39-14-34 całodobowo
automatyczna sekretarka – prosimy zostawić wiadomość, skontaktujemy się!

Główni wytwórcy komputerów osobistych

Już od kilku lat najbardziej dynamicznym w rozwoju działem systemów komputerowych są komputery osobiste. Tak było również w 1992 r., kiedy to niektórym firmom udało się podwoić wartość sprzedaży, a wśród firm, które pozostawały w pierwszej pięćdziesiątce producentów komputerów osobistych, zarówno w 1991, jak i 1992 r. przeważały te, które poprawiły swą pozycję.

W tabeli 1 przytaczamy za pismem Datamation z grudnia 1993 r. [1] listę pierwszych pięćdziesięciu wytwórców w 1992 r. według wartości sprzedanych komputerów osobistych, podając jednocześnie ich pozycje oraz wartość sprzedaży w analogicznym rankingu z 1991 r., zmianę procentową oraz kwoty uzyskane ze sprzedaży w pierwszym półroczu 1993 r. Dla porównania podano też liczbę komputerów (w tys. szt.) sprzedanych w latach 1991 i 1992 oraz odpowiadające tym liczbom zmiany procentowe.

Wskaźniki rozwoju, jakie osiągnęły czołowe firmy są lepsze niż w latach ubiegłych, a kwota uzyskana przez wszystkie 50 firm (48,8 mld dolarów) też jest nowym rekordem. Jednak łączny wzrost wartości sprzedaży 50 najlepszych firm wyniósł tylko 5,6%, co jest rezultatem ostrej walki cenowej ograniczającej zyski. Na przykład, firma Apple, która wykazywała 10% wzrostu wpływów w 1992 r. miała w pierwszym kwartale tego roku 44% marżę zysku. Natomiast w 1993 r. mimo wzrostu sprzedaży o 21%, zysk zmniejszył się do 25%. Jeszcze dobitniej widać to na przykładzie firmy Dell, która w 1992 r. osiągnęła wpływy ponad 1,8 mld dolarów, oznaczające wzrost o ponad 170% w stosunku do 1991 r. Jeśli chodzi o liczbę sprzedanych komputerów, wzrost ten był jeszcze większy i wyniósł aż 190%. Jednakże marża zysku obniżyła się z 26% w 2. kwartale 1992 r. do zaledwie 6,5% w IV kwartale. Sytuacja taka spowodowała kłopoty finansowe takich firm, jak Everex i Tandon.

Ceny komputerów osobistych spadły gwałtownie. Jeszcze niedawno sięgały one 10 tys. dolarów, a obecnie oparte na procesorze 486 takie systemy, jak ValuePoint IBM, Dimension Della lub ProLinea firmy Compaq, są sprzedawane poniżej 2500 dolarów. Firmy takie, jak Dell i Gateway 2000 znalazły się po raz pierwszy w pierwszej dziesiątce, eliminując z niej firmy AT & T/NCR, Commodore i Unisys.

Pozostająca w 1992 r. na pierwszej pozycji IBM straciła dystans do swoich rywali i w pierwszej połowie 1993 r. została wyprzedzona przez Apple i Compaq. Próbuje ona ratować sytuację tworząc podległe sobie organizacje sprzedaży komputerów osobistych, takie jak IBM PC Direct i Ambra, które przejęły znaczną część tej roli.

Tabela 2 pokazuje 20 firm o największych wskaźnikach wzrostu sprzedaży w 1992 r. w stosunku do 1991 r. wraz z procentową wartością tych wskaźników.

Wśród występujących tu firm warto zwrócić uwagę na Digital, który ostatnio podejmuje energiczne wysiłki w sprzedaży komputerów osobistych. Wiadomo też, że podobną akcję prowadzi Hewlett-Packard.

W mikroprocesorach stosowanych w komputerach osobistych dominuje rodzina X86 Intela, która w 1993 r. stanowiła 87% wszystkich sprzedanych mikroprocesorów [2]. Z rodziny

Tabela 1. Pierwszych pięćdziesięciu producentów komputerów osobistych

Kolejność		Firma	Sprzedaż		Zmiana	Sprzedaż	Liczba		Zmiana
1992	1991		1992	1991		w I poł. 93r	1992	1991	
			mln $USA		[%]	mln $USA	tys. szt.		[%]
1	1	IBM	7655	8505	-10,0	2915	3306	3058	8,1
2	3	Apple	5412	4900	10,5	3836	2910	2261	28,7
3	4	Compaq	4100	3271	25,3	3244	1560	998	56,3
4	2	NEC	3987	5280	-24,5	2221	1406	1438	-2,2
5	5	Fujitsu	2618	2342	11,8	1414	230	190	21,2
6	6	Toshiba	1949	2045	-4,7	974	509	566	-10,1
7	18	Dell	1813	667	171,6	1373	774	267	190,0
8	7	Olivetti	1349	1578	-14,5	527	631	563	12,1
9	13	AST Research	1141	827	37,9	780	437	251	55,7
10	25	Gateway 2000	1107	627	76,6	786	437	251	74,3
11	9	AT&T/NCR	999	950	5,2	493	145	216	-32,7
12	12	Hitachi	892	825	8,1	497	19	29	-34,5
13	16	Acer	890	709	25,5	502	588	476	23,5
14	19	Packard Bell	879	641	37,1	514	700	515	36,0
15	17	Seiko Epson	741	708	4,8	400	213	292	-27,2
16	20	Canon	741	613	21,0	488	6	13	-53,9
17	8	Commodore	732	1039	-30,0	313	1195	1194	0,1
18	15	Hewlett-Packard	725	715	1,4	310	302	177	71,1
19	33	ICL	713	297	139,8	604	202	-	-
20	-	Vobis	686	-	-	-	396	-	-
21	11	Groupe Bull	686	890	-22,9	266	495	468	5,7
22	14	Tandy	650	750	-13,3	284	571	477	19,7
23	21	Siemens Nixdorf	526	362	45,4	339	229	178	28,7
24	23	CompuAdd	524	513	2,2	-	206	141	46,0
25	39	Mitac	522	441	18,4	274	204	194	5,3
26	26	Digital	450	215	109,3	450	203	61	232,7
27	10	Unisys	434	900	-51,8	186	152	96	58,7
28	50	Northgate	400	167	139,3	-	-	-	-
29	36	Hyundai Electron.	391	269	45,6	213	235	304	-22,8
30	47	Tatung	366	282	29,5	151	660	625	5,6
31	31	Everex	350	307	13,9	-	-	201	-
32	27	Ricoh	340	409	-16,9	204	35	39	-9,1
33	38	Mitsubishi	311	242	28,2	-	-	-	-
34	45	First Internat.	301	204	47,4	184	-	-	-
35	41	Trigem	286	217	31,8	-	-	-	-
36	24	Tandon	285	461	-38,3	-	132	190	-30,4
37	48	Samsung	267	199	34,1	-	-	-	-
38	43	Memorex Telex	257	255	0,7	97	120	70	70,7
39	-	Compudyne	238	-	-	-	-	-	-
40	-	Copam	225	-	-	-	150	-	-
41	28	Wang	220	340	-35,3	-	104	107	-3,0
42	46	Matsushita	215	203	6,1	-	-	-	-
43	40	Zeos	206	231	-10,7	113	-	-	-
44	35	Amstrad	202	272	-25,5	87	220	254	-13,3
45	37	ALR	200	227	-12,1	80	120	142	-14,9
46	42	Tulip Computers	189	215	-12,1	101	97	112	-13,6
47	-	Goldstar	182	194	-6,4	88	-	-	-
48	-	Leading Edge	170	90	88,9	85	125	72	73,6
49	-	Texas Instruments	160	150	6,7	121	65	47	38,1
50	-	Xerox	144	-	-	-	-	-	-

tej najwięcej było procesorów 486DX (prawie 37%), a następnie 486SX (niespełna 33%) i 386 (prawie 30%). Procesorów Pentium sprzedano w 1993 r. tylko 210 tysięcy, co stanowiło zaledwie 0,5% rodziny X86. Według przewidywań firmy Mercury Research w 1994 r. proporcje te ulegną zmianie i przy ogólnym wzroście sprzedanych mikroprocesorów o 18,9% do 53,1 mln sztuk, wzrost procesorów X86 będzie jeszcze większy (21,4%), a wśród nich największy wzrost (ponad 10-krotny) wykaże Pentium, którego sprzedaż wyniesie 2,13 mln sztuk (4,6% rodziny X86). O ponad 50% wzrośnie sprzedaż 486SX (do 19,2 mln – 40,6% X86), ale najliczniejszym typem będzie nadal procesor 486DX (19,5 mln – 41,3%), mimo wolniejszego tempa wzrostu sprzedaży (poniżej 36%).

Tabela 2. Firmy o największym przyroście sprzedaży w latach 1991–1992

L. p.	Nazwa firmy	Wskaźnik wzrostu [%]
1	Dell	171,6
2	ICL	139,8
3	Northgate	139,3
4	Digital	109,3
5	Leading Edge	88,9
6	Gateway 2000	76,6
7	First International	47,4
8	Hyundai Electronics	45,6
9	Siemens Nixdorf	45,4
10	AST Research	37,9
11	Packard Bell	37,1
12	Samsung	34,1
13	Trigem	31,8
14	Tatung	29,5
15	Mitsubishi	28,2
16	Acer	25,5
17	Compaq	25,3
18	Canon	21,0
19	Mitac	18,4
20	Everex	13,9

Według tych samych źródeł 1995 r. ma przynieść dalsze zmiany. Wzrost sprzedaży w stosunku do 1994 r. ma być niższy (10,7% ogólnie i 6,5% dla X86). W większym zakresie wykorzystywać się będzie mikroprocesory RISC, takie jak PowerPC triumwiratu Apple/IBM/Motorola, Alpha Digitala i układu Mips firmy Silicon Graphic. Jednakże procesory X86 stanowić będą nadal 88,6% wszystkich mikroprocesorów, a więc więcej niż w 1993 roku i tylko o 0,3% mniej niż w roku 1994. Tak samo największy rozwój wykaże Pentium, ale tym razem już tylko niespełna trzykrotny w stosunku do poprzedniego roku (5,85 mln co stanowi 11,6% X86). Sprzedaż procesora 486DX ma wzrosnąć o 46,4% do 28,6 mln i stanowić on ma aż 56,8% wszystkich X86. Natomiast spadać już będzie sprzedaż zarówno 486SX (o 31% do 13,2 mln), jak i 386 (o 58,3% do 2,7 mln). Stanowić one wówczas mają tylko, odpowiednio 26,3% i 5,3% wszystkich procesorów X86.

LITERATURA

[1] Mike Ricciuti: The PC 50, Datamation December 1, 1993, pp. 36–8
[2] Market Forecast 1994, Electronics 10 January 1994, pp. 5–14.

JAN RYŻKO

Międzynarodowe Targi Zastosowań Informatyki w Przemyśle i CAD/CAM '94 odbędą się w Warszawie w Pałacu Kultury i Nauki w dniach 7–10 czerwca 1994 r.

Organizuje je w salach Pałacu Kultury i Nauki już po raz drugi Ministerstwo Przemysłu i Handlu wraz z Biurem Reklamy SA.

Targi mają na celu przybliżenie polskim odbiorcom, a szczególnie przedstawicielom polskiego przemysłu tematyki zastosowań informatyki, automatyki, teletransmisji i nowoczesnych technik organizacji pracy w przemyśle oraz komputerowego wspomagania projektowania.

Organizatorzy za cel główny, oprócz prezentacji najnowszych osiągnięć informatyki światowej, stawiają sobie stworzenie odpowiednich warunków sprzyjających nawiązaniu bezpośrednich kontaktów między przedstawicielami polskiego przemysłu, zainteresowanymi komputeryzacją procesów produkcyjnych, a firmami prezentującymi swoją najnowszą ofertę w tej dziedzinie.

Powodzenie poprzedniej imprezy oraz duże zainteresowanie jakim cieszyła się w środowisku informatycznym i kręgach przemysłowych pozwala sądzić, że skala tegorocznej imprezy, jeśli chodzi o powierzchnię wystawienniczą i liczbę uczestników, będzie znacznie większa.

Międzynarodowym Targom Zastosowań Informatyki w Przemyśle i CAD/CAM '94 towarzyszyć będzie, podobnie jak w roku ubiegłym, cykl konferencji i seminariów organizowanych przez Centrum Informacji Naukowej i Przemysłowej a prowadzonych przez wybitnych specjalistów.

Redakcja nasza przygotowuje czerwcowy numer INFORMATYKI związany tematycznie z tymi Targami. Oprócz artykułów merytorycznych omawiających zastosowanie informatyki w przemyśle, stanowiących trzon numeru, zamierzamy zaprezentować wystawców biorących udział w imprezie.

Zapraszamy Firmy do zamieszczenia w tym numerze ogłoszeń o swoich ekspozycjach i produktach, a także wkładek reklamowych i tekstów sponsorowanych.

Wszelkich informacji udziela:
redakcja INFORMATYKI, tel. 39-14-34, 01-552 Warszawa, Pl. Inwalidów 10, p. 104

Redakcja INFORMATYKI przygotowuje raport nt. sieci komputerowych i ich wykorzystania w polskich przedsiębiorstwach. Omówienie ankiety zostanie opublikowane na naszych łamach. Wszystkich zainteresowanych prosimy o wypełnienie ankiety i przesłanie faksem pod nr (2) 642-81-38 lub pod adresem naszej redakcji.

Nazwa firmy _____

Działalność _____

Adres _____

Miasto _____

Kod pocztowy _____

Telefon _____

Fax _____

Imię i nazwisko _____

Stanowisko _____

1. Host IBM 3090 ☐ IBM 3080 ☐ IBM 4300 ☐ Inne ☐

2. Mini AS/400 ☐ Unisys ☐ UNIX ☐ VAX ☐ S/3x ☐ Inne _____

3. Liczba sieci lokalnych _____

4. System zarządzania w sieci
   NOVELL ☐ IBM LAN Serv. ☐
   LAN MNGR ☐ PATHWORKS ☐
   BANYAN ☐ Inne _____

5. Rodzaj sieci
   ETHERNET ☐ TOKEN RING ☐
   COAX ☐ TP ☐
   Inne ☐

6. System
   Attachmate ☐ Eicon ☐
   IBM ☐ Novell ☐
   DCA ☐ Inne _____

7. Oprogramowanie emulacji stacji roboczej
   Attachmate ☐ Eicon ☐
   Rumba ☐ IBM PC/3270 ☐
   DCA ☐ Inne _____

8. Liczba PC włączonych do sieci _____

9. System operacyjny PC DOS ☐ WINS ☐ OS/2 ☐ Mac ☐ UNIX ☐

10. Liczba stacji roboczych w sieci _____

11. Rodzaj stacji roboczych 3270 ☐ 5250 ☐ Inne _____

12. Liczba użytkowników PC lub stacji roboczych _____

13. Czy jest połączenie z siecią zewnętrzną? Tak ☐ Nie ☐

14. Typ połączenia
   Linia punkt za punktem ☐ X.25 ☐
   POLPAK ☐ Inne ☐

15. Jakiego okresu dotyczy planowanie inwestycyjne w Waszej firmie? 3–6 miesięcy ☐ 6–12 miesięcy ☐

Refleksje

Przebieg oraz ostateczne wyniki tegorocznych Targów CeBIT'94 dowiodły raz jeszcze ich miażdżącej przewagi nad imprezami konkurencyjnymi. Przedwczesne okazały się prognozy, że CeBIT zacznie tracić swą popularność i dominację właśnie wskutek swych monstrualnych rozmiarów, przerastających możliwości percepcyjne przeciętnego człowieka. Okazało się, że Targi zdołały w pełni obronić swą dotychczasową formułę, czego niepodważalnym dowodem jest ich nieprzerwany rozwój bez najmniejszych nawet objawów załamania, czy nawet stagnacji. Realizowane w ostatnich latach systematyczne porządkowanie oraz wyraźne tematyczne grupowanie ekspozycji, a także intensywne inwestowanie w infrastrukturę Targów sprawiły, że CeBIT jest w skali światowej nadal bezkonkurencyjny, wpływając w odczuwalny sposób na zmniejszenie się atrakcyjności innych imprez międzynarodowych o podobnym charakterze.

Dowodzi tego kolejny wzrost dwóch podstawowych wskaźników, jakimi są: liczba wystawców oraz liczba zwiedzających. W CeBIT'94 uczestniczyło ostatecznie 5850 wystawców z 54 krajów (1993 r. – 5614), w tym 2280 zagranicznych (1993 r. – 2190), a liczba zwiedzających osiągnęła niewiarygodny wprost pułap 675 000 osób (1993 r. – 660 700), w tym 105 000 z zagranicy (1993 r. – 101 800). Sukces ten jest tym bardziej wart podkreślenia, że trwająca od kilku lat zachodnioeuropejska recesja, zwłaszcza w RFN, nie została jeszcze całkowicie przezwyciężona i jest nadal silnie odczuwana zarówno w informatyce, jak i telekomunikacji.

Atrakcyjność CeBIT najlepiej ilustruje średnia liczba zwiedzających, która w br. przekroczyła 84 000 osób, przy czym w dniu 17 marca padł nawet rekord dzienny frekwencji: 105 000 zwiedzających! Uciążliwość pracy non-stop personelu stoisk wystawowych (łącznie ok. 80 000 osób) oraz związane z tym wysokie koszty uczestnictwa w Targach spowodowały, że niektórzy wystawcy wysunęli ostatnio żądanie skrócenia czasu trwania ekspozycji do 6 dni. Przeciwstawili się temu jednak zwiedzający, których część w przeprowadzonej ankiecie żądała nawet wydłużenia imprezy o dodatkowe dni. W tej sytuacji organizatorzy postanowili również na przyszły rok utrzymać dotychczasowy okres 8-dniowy, uzasadniony przepustowością terenów targowych oraz możliwościami zakwaterowania i komunikacji miejskiej.

Oceniając merytoryczną treść ekspozycji należy stwierdzić, że mimo coraz ostrzejszej konkurencji i oszałamiającego postępu technicznego, na tegorocznym CeBIT nie pojawiły się produkty o charakterze przełomowym. Na każdym kroku można było napotkać dowody systematycznego udoskonalania dotychczasowych produktów, wyrażające się w znacznej poprawie najważniejszych parametrów eksploatacyjnych i użytkowych. Najbardziej spektakularnym dowodem tego rodzaju postępu była dziedzina technik multimedialnych. Istotny postęp widoczny był w dziedzinie mikroprocesorów (Power PC), który sprawia, że komputery osobiste przekształcają się w stacje robocze, zdolne zaspokoić najbardziej wyrafinowane potrzeby użytkowników. Niewątpliwą nowością jest pojawienie się sprzętu mikrokomputerowego typu PDA (*Personal Digital Assistent*), reprezentującego dalszy postęp w kierunku miniaturyzacji komputerów przenośnych. Ze względu na rosnące wymagania użytkowników w rozwoju sprzętu i oprogramowania zaobserwować można stały trend w kierunku wzrostu łatwości użytkowania a także coraz korzystniejsze relacje wydajności do ceny.

Tobiasz J.: Modelowanie powierzchni siatkowych w systemie AutoCAD INFORMATYKA 1994, nr 5, s. 2 Metoda wykorzystania systemu AutoCAD do automatyzacji modelowania wzorców kształtów naturalnych, które mogą być wykorzystywane w symulacjach.	**Tobiasz J.: Netzoberflächenmodellieren im AutoCAD-System** INFORMATYKA 1994, Nr. 5, S. 2 Eine Methode für Ausnutzung des AutoCAD--Systems zur Automatisierung des Modellierenprozesses von Naturalformenmustern, die für Simulationen ausgenutzt werden können.	**Tobiasz J.: Mesh surfaces modelling in the AutoCAD system** INFORMATYKA 1994, No. 5, p. 2 Method of the AutoCAD system utilization for automatized modelling of natural shape patterns, which can be used for simulations.
Czajkowski G., Zieliński K.: PCN – system rozproszonego przetwarzania INFORMATYKA 1994, nr 5, s. 8 Charakterystyka języka PCN (Program Composition Notation) do szybkiego projektowania i implementacji programów równoległych.	**Czajkowski G., Zieliński K..: PCN – ein System für verteilte Datenverarbeitung** INFORMATYKA 1994, Nr. 5, S. 8 Eine Charakteristik der PCN (Program Composition Notation) – Sprache für schnelle Projektierung und Implementierung von Parallelprogrammen.	**Czajkowski G., Zieliński K.: PCN – a distributed processing system** INFORMATYKA 1994, No. 5, p. 8 Characteristics of the PCN (*Program Composition Notation*) language for fast designing and implementing of parallel programs.
Miłosz M., Pranagal R.: Budowa prototypu systemu ekspertowego INFORMATYKA 1994, nr 5, s. 14 Charakterystyka procesu budowy prototypu systemu ekspertowego na przykładzie rozdzielni wysokiego napięcia w wielkim zakładzie przemysłowym.	**Miłosz M., Pranagal R.: Bau eines Expertensystemprototyps** INFORMATYKA 1994, Nr. 5, S. 14 Eine Charakteristik vom Bauprozess eines Expertensystemprototyps auf Beispiel einer Hochspannungsschaltanlage in einem Industriegrossbetriebe.	**Miłosz M., Pranagal R.: Expert system prototype building** INFORMATYKA 1994, No. 5, p. 14 Characteristics of an expert system prototype building process on example of high voltage switching-station in a big industrial enterprise.
Czarnacka-Chrobot B.: Metoda FUNCTION POINTS INFORMATYKA 1994, nr 5, s. 19 Charakterystyka metody oceny pracochłonności wykonania projektów systemów informatycznych oraz badania produktywności pracy projektantów takich systemów.	**Czarnacka-Chrobot B.: Die FUNCTION POINTS-Methode** INFORMATYKA 1994, Nr. 5, S. 19 Eine Charakteristik von Methode zur Arbeitsaufwandschätzung der Projekterarbeitung eines EDV-Systems und zur Forschung der Arbeitsproduktivität von Projektanten solcher Systeme.	**Czarnacka-Chrobot B.: The FUNCTION POINTS method** INFORMATYKA 1994, No. 5, p. 19 Characteristics of the method for labor consumption evaluation of information system design and for research of labor productivity of such system designers.

Bardzo interesującym, a u nas jeszcze niedocenianym elementem wielu eksponatów było zastosowanie rozwiązań prowadzących do zmniejszenia zużycia energii przez komputery klasy PC. Rozwiązania takie są już powszechnie stosowane w komputerach klasy laptop i notebook, w których ograniczny czas eksploatacji źródeł zasilania (akumulatory) wymusił zastosowanie układów automatycznie wyłączających komputer w czasie, gdy nie jest on efektywnie wykorzystywany. Na CeBIT'94 po raz pierwszy demonstrowano tzw. energooszczędne PC. Okazało się, że już ponad 100 producentów tej klasy sprzętu zastosowało się do wytycznych amerykańskiej Agencji Ochrony Środowiska EPA (*Environmental Protection Agency*) ze względu na fakt, że od października ub.r. wszystkie placówki administracji państwowej USA mogą kupować wyłącznie komputery odpowiadające tym wytycznym.

Komputery, a zwłaszcza monitory, w czasie, gdy nie są wykorzystywane, powinny automatycznie przełączać się na tzw. energooszczędny tryb pracy, w którym zużycie energii elektrycznej nie może być większe niż 30 W. Wytyczne te dotyczą również drukarek.

Badania bowiem wykazały, że aż ok. 40% wszystkich amerykańskich PC eksploatuje się w wymiarze ponad 3000 godzin rocznie, co przy średniej mocy jednego komputera 0,438 kW powoduje zużycie aż 1314 kWh. Tymczasem komputery te efektywnie wykorzystywane są zaledwie 600 godzin rocznie (20% czasu włączenia), a więc powinny zużywać tylko ok. 260 kWh. Wyliczono, że wprowadzenie wspomnianych rozwiązań we wszystkich użytkowanych w USA komputerach PC pozwoliłoby zaoszczędzić rocznie ok. 2 mld dolarów, a więc wyłączyć z eksploatacji kilka zatruwających środowisko bloków energetycznych. Eksponowane na CeBIT modele energooszczędne miały umieszczony na obudowie specjalny znaczek „Energy Star" wspomnianej Agencji. Spełnienie warunków EPA osiągnęto przez rozszerzenie funkcji procesora o dodatkowe układy sterujące, które w sposób ciągły obserwują pobór mocy przez poszczególne moduły komputera i wyłączają je na czas, gdy nie są wykorzystywane.

Oczywiście na tle naszego gigantycznego marnotrawstwa energii wspomniane wysiłki amerykańskie, wyrażające zresztą tendencje ogólnoświatowe, mogą mieć u nas posmak prawie anegdotyczny, ale powinny przyczyniać się do stopniowego przekształcania naszej świadomości i przyzwyczajeń.

Na CeBIT eksponatom i wystawcom nie są przyznawane wyróżnienia, jak to czynią organizatorzy wielu imprez targowych. Aby usunąć ten mankament, znane niemieckie wydawnictwo czasopism informatycznych Vogel Verlag (wydawca m.in. miesięcznika CHIP) począwszy od ub.r. przyznaje w dziedzinie komputerów osobistych wyróżnienia o nazwie CeBIT Highlights. 40-osobowe jury, złożone z najlepszych ekspertów (w tym specjalistów z laboratorium CHIPA) wytypowało i przyznało wspomniane wyróżnienia w 11 różnych kategoriach sprzętu i oprogramowania. Szczegółowe omówienie wyróżnionych produktów postaramy się zamieścić w następnym numerze INFORMATYKI. Sądzę, że prezentacja tych wyróżnień przekaże bardziej obiektywny obraz najciekawszych pozycji CeBIT'94 niż mogłaby to uczynić najliczniejsza nawet grupa ekspertów polskich.

WŁADYSŁAW KLEPACZ

PIERWSZY POLSKI INTELIGENTNY
SYSTEM INFORMATYCZNY WSPOMAGAJĄCY ZARZĄDZANIE
PRODUKCJĄ - (TPPiRP), FINANSAMI - KOSZTAMI, SPRZEDAŻĄ, KADRAMI
WRAZ Z SYSTEMEM INFORMACJI KIEROWNICTWA
WSPOMAGANY SYSTEMEM SZTUCZNEJ INTELIGENCJI PC-SHELL
ORAZ EKSPERCKĄ BAZĄ WIEDZY EKONOMICZNO-FINANSOWEJ - ISAF
(DLA DUŻYCH PRZEDSIĘBIORSTW PRODUKCYJNYCH)

Perfect-Ekspert AITECH

(DZIAŁAJĄCY W OPARCIU O ŚRODOWISKO PRACY - WYŁĄCZNIE SYSTEM OPERACYJNY DOS, NetWare v3.11, v4.01 FIRMY NOVELL)

OFERUJE

PRZEDSIĘBIORSTWO PRODUKCYJNO-USŁUGOWE INNOWACJI TECHNICZNYCH

JUNISOFTEX Sp. z o.o.

ROK ZAŁOŻENIA 1987
WARTOŚĆ SPRZEDAŻY wg F-02 1992 (14,6 mld), wg F-01 na 1993.10.31 (15,4 mld)
TEL/FAX 315-131, 315-700, 321-525, 388-052, 388-054 TLX 036233
BIURO TECHNICZNO-HANDLOWE
44-100 GLIWICE ul. Kościuszki 1
WSZYSTKICH ZAINTERESOWANYCH ZAPRASZAMY NA MERYTORYCZNE ROZMOWY DO NASZEGO BIURA TECHNICZNO-HANDLOWEGO
OFERUJEMY RÓWNIEŻ INTELIGENTNE NARZĘDZIE DO PRODUKCJI EKSPERCKICH BAZ WIEDZY

PC-SHELL

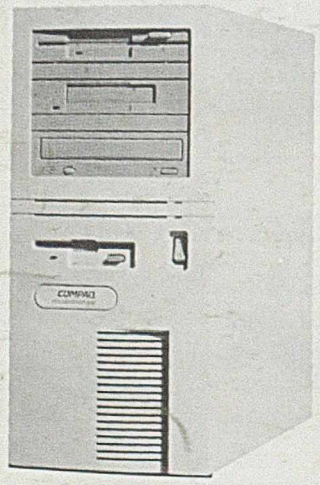
ARTIFICIAL INTELLIGENCE

DLA JEDNOSTEK BUDŻETOWYCH (NAUKA, OŚWIATA, URZĘDY MIAST I GMIN, SŁUŻBA ZDROWIA) BEZ PRAWA DO KOMERCYJNEGO WYKORZYSTYWANIA W CENIE 5.000.000,- (BEZ VAT) - INFORMACJE I ZAMÓWIENIA PROSIMY SKŁADAĆ BEZPOŚREDNIO U PRODUCENTA ARTIFICIAL INTELLIGENCE LABORATORY - AITECH, KATOWICE 14 SKR. POCZT. 2804.
DLA POZOSTAŁYCH JEDNOSTEK GOSPODARCZYCH BEZ PRAWA KOMERCYJNEGO WYKORZYSTYWANIA TJ. WYŁĄCZNIE NA WŁASNE POTRZEBY DO KOŃCA I KW. 1994 ROKU W CENIE PROMOCYJNEJ 10.000.000,- (BEZ VAT), DLA KUPUJĄCYCH DO CELÓW KOMERCYJNYCH DO KOŃCA I KW. 1994 25.000.000,- (BEZ VAT), LICENCJA OGRANICZONA NA NIEOKREŚLONĄ ILOŚĆ ODBIORCÓW KOŃCOWYCH.
POSZUKUJEMY DO WSPÓŁPRACY LUB NAWIĄŻEMY WSPÓŁPRACĘ Z EKSPERTAMI Z DOWOLNYCH DZIEDZIN NAUKI I TECHNIKI.

0/3/93

Serwer już za 3300 USD

Compaq Computer Corporation, który konsekwentnie już od ponad 2 lat obniża ceny swoich produktów, wprowadził w marcu na rynek nową rodzinę serwerów Compaq ProSignia SV – ulepszoną wersją cenionej rodziny serwerów Compaq ProSignia – których cena nie przekracza ceny PC-tów. Wszystkie modele z tej serii, zgodnie z wymaganiami EISA, są wyposażone standardowo w procesor 486SX/33, 486DX/33 lub 486DX2/66. Modele standardowe są dostarczane również z 8 lub 16 MB pamięci RAM, zintegrowanym sterownikiem NexFlex-L ENET na szynie lokalnej, lub nowym sterownikiem NetFlex-2 TR (dla Token Ring) w slocie EISA, zintegrowanym 32-bitowym sterownikiem Fast-SCSI-2 na szynie lokalnej, pięcioma gniazdami rozszerzającymi dla magistrali EISA i pięcioma kieszeniami na pamięci masowe.



Użytkownicy mają też do wyboru modele z dyskiem twardym o pojemności 1,05 GB lub z nowym napędem 535 MB. Compaq ProSignia SV wykorzystują Compaq SmartStart, inteligentny procesor (wymagający napędu CD) upraszczający instalację systemów operacyjnych, takich jak NetWare, Microsoft Windows NT czy SCO UNIX. Zapewnia on też niezawodną konfigurację. Modele Compaq ProSignia SV są wyposażone w nową edycję oprogramowania – Compaq Insight Manager, drivery Ethernet (pełny duplex) oraz ARCserve v. 5.1. – Windows Edition. Przygotowano również nową, wysokowydajną opcję rozszerzeniową – SCSI Storage Expander – zewnętrzny moduł z dwiema kieszeniami SCSI o połowie wysokości (ang. half-height) na twarde dyski, napędy taśmy lub CD-ROM.

Ta nowa rodzina serwerów jest przeznaczona dla niewielkich zespołów roboczych, wydziałów większych jednostek organizacyjnych oraz małych i średnich firm. Wszystkie modele są objęte trzyletnią gwarancją. W Europie serwery z serii Compaq ProSignia SV można nabyć już za ok. 3300 USD. Oczywiście, trzeba pamiętać, że u nas dojdzie jeszcze cło i podatek VAT.

Drukarka IBM 4037 5E firmy Lexmark

Na rynku drukarek komputerowych największym powodzeniem cieszą się obecnie drukarki nieuderzeniowe: laserowe, atramentowe oraz drukujące przy użyciu techniki LED. Jest to spowodowane m.in. ich dużymi możliwościami graficznymi (co zasadniczo wpływa na jakość wydruku) oraz cichą pracą. Szczególnie jeden z rodzajów drukarek nieuderzeniowych przeżywa obecnie okres znaczącej prosperity (sprzedaż rośnie w tempie 14,5% rocznie) – są to drukarki typu LED, pracujące z szybkością od 1 do 10 stron/minutę.

Drukarka IBM 4037 5E firmy Lexmark należy właśnie do tej grupy. Jej podstawową cechą jest zastosowana w niej technologia wydruku. Podobna jest ona do tej, którą stosuje się w drukarkach laserowych, gdzie obraz strony jest „rysowany" na światłoczułym bębnie (podobne pracują w kserokopiarkach) promieniem laserowym, odbijanym ruchomym układem luster. W drukarce IBM 4037 5E rolę lasera pełni rząd diod świecących typu LED, umieszczonych w jednym rzędzie, z gęstością taką, jak rozdzielczość druku.

Niezawodność każdego urządzenia jest określona przez parametr MTBF (Mean Time Between Failure) pokazujący średni czas pracy bezawaryjnej; dla drukarki IBM 4037 5E wynosi on 25 tys. godz. pracy. Gwarantowana, maksymalna wynosi 16 tys. stron/mies. Obie te wartości świadczą dobitnie o klasie oferowanego urządzenia.

Drukarka IBM 4037 5E drukuje dokumenty z szybkością 55 stron/minutę. Otrzymany dzięki niej wydruk charakteryzuje się rozdzielczością 300 × 300 dpi, a przy zastosowaniu specjalnej techniki polepszania obrazu PQET (Print Quality Enhabcement Technology) aż 1500 × 300 dpi. Technika PQET została opracowana przez IBM i w skrócie polega na zmniejszaniu wymiarów kropeczek występujących na krawędziach litery czy innego znaku. Oznacza to, że za niewielką cenę można uzyskać wydruk o jakości, jaka jest możliwa do uzyskania na drogim sprzęcie laserowym.

Kolejna innowacja, jaką stosuje Lexmark w swoim sprzęcie, to użycie kart z pamięcią „flash". Karta typu SIMM z pamięcią „flash" ma pojemność od 256 do 1 MB, cały obszar pamięci RAM w drukarce wynosi 512 kB – z możliwością rozszerzenia do 4 MB. IBM 4037 5E firmy Lexmark jest wyposażona m.in. w sterowniki Windows 3.0 i 3.1, WordPerfect 5.1, Lotus 1-2-3 do wersji 3.4, Word 5.5, Works 2.0 i wiele innych. Zawiera wiele typów fontów bitmapowych (24) i wektorowych (16, Type 1, dodatkowo 23 typy fontów skalowalnych na karcie SIMM). Emuluje pracę urządzeń IBM LaserPrinter 4019/4029, Proprinter 4202 i HP LaserJet II. Zawiera pojemnik na 150 arkuszy A4 drukuje na papierze, kopertach, etykietkach, foliach. Zawartość pojemnika na toner wystarcza na zadrukowanie 3500 arkuszy. Pojemnik ten może być regenerowany, co jest bardzo korzystne z ekologicznego punktu widzenia. W czasie pracy zużywa 200 W, przy czym „czuwając" pobiera tylko 12 W. W trakcie procesu drukowania nie wytwarza się ozon (jak to występuje w wielu drukarkach laserowych).

Instalacja drukarki jest bardzo prosta. Odbywa się pod nadzorem specjalnie w tym celu opracowanej aplikacji, QuickStart, uruchomionej na komputerze połączonym z drukarką. Program ten przegląda zawartość pamięci masowej PC i zapytuje użytkownika, czy zainstalować odpowiednie sterowniki. W porównaniu z konkurentkami w swojej klasie jest to drukarka o lepszej m.in. jakości wydruku, a dodatkowo pozwala na bardziej intensywną, niż jej rywalki, pracę. Może współpracować z komputerami IBM PC, AT, PS/2 za pomocą łącza równoległego. Istnieje możliwość pracy z urządzeniami RISC Systems/600 i AS/400. Jej rozmiary wynoszą 38 × 45 × 24,5 cm, a ciężar 13 kg.

Nowe narzędzia SMCC

Sun Microsystems Computer Corporation rozpoczęła w styczniu br. dostawy nowego zintegrowanego pakietu narzędzi do rozwoju oprogramowania rdzeniowego, stosowanego do oferowanych przez firmę unixowskich aplikacji klient-serwer. Zestaw ten, nazwany SunPro WorkShop, umożliwia programistom szybsze opracowywanie i wdrażanie aplikacji. W skład zestawu wchodzi pełna gama narzędzi programistycznych dla platformy SPARC/Solaris. Zestaw SunPro WorkShop jest oferowany w trzech konfiguracjach (C, C++ i FORTRAN). Wymaga środowiska operacyjnego Solaris 2.2 lub nowszego.

W ostatnim okresie Sun odnosi spore sukcesy. Czytelnicy amerykańskiego pisma Control, poświęconego automatyce przemysłowej, uznali system SPARCstation 10 za najlepszą stację roboczą używaną do automatycznego sterowania produkcją. Na drugim miejscu znalazły się ex aequo IBM i Hewlett-Packard. Natomiast w rankingu niemieckiego czasopisma związanego z biznesem – Manager Magazin – Sun został uznany za odnoszącą największe sukcesy i najlepiej zarządzaną firmę komputerową na świecie. W badaniach prowadzonych przez specjalistów tego pisma i przez Boston Consulting Group, Sun zajął dziewiąte miejsce w rankingu najlepiej radzących sobie firm notowanych na giełdach w 14 najbardziej uprzemysłowionych krajach świata. Wśród firm amerykańskich, w tym samym rankingu, Sun zajął czwarte miejsce, za Home Depot, Intelem i Motorolą.

Warto przy okazji przypomnieć, że Sun Microsystems Inc., którego częścią jest SMCC – jest wyłącznym dostawcą komputerów na potrzeby Mistrzostw Świata w Piłce Nożnej w 1994 r.

Opracowała:
KRYSTYNA
KARWICKA-RYCHLEWICZ